Cookbook 5: How To Serve a GeoSciML Version 3.2 Web Feature Service (WFS) Using GeoServer, Version 1.1

DOCUMENT CONTROL

Change Summary

Version	Date	Author/Editor	Change Summary
1.0	25th April 2014	Marcus Sen	
1.1	12th September 2014	Marcus Sen	Fix links in PDF and formatting marks

Contents

Introduction

This guidance describes how to set up a web service server using free and open source Geoserver software to provide a OneGeology compliant ISO/OGC Web Feature Service (WFS) version 2.0 serving GeoSciML v3.2 as described in OneGeology Cookbook 4: GeoSciML 3.2 Encoding Cookbook for OneGeology 5 star WFS services. Version 1.0. Whilst a OneGeology WFS does not need to be INSPIRE compliant, INSPIRE WFS are simply standard WFS 2.0 and therefore this guidance can be used with the proviso that the sections on INSPIRE extended capabilities need not be provided for OneGeology purposes but can be if you want. It describes how you can implement the WFS functionality described in chapters 6 and 7 of the latest Technical Guidance for the implementation of INSPIRE Download Services. A WFS setup using the encoding guidance in Cookbook 4 will enable the service to be used and queried using the OneGeology portal age and lithology WFS query tool, thus enabling the service to achieve 5 star accreditation.

This guidance includes example data and configuration files that can be used to create running Open GeoSpatial Consortium (OGC) Web Feature Services (WFS) that deliver data in GeoSciML. This example illustrates aspects of service and data configuration that should help the reader to set up services using their own data.

Pre-requisites / System Requirements

The cookbook is technical and some assumptions are made about the reader's background knowledge:

- The reader is a, or is working closely with an expert in the data model and schema for GeoSciML who has read the sister cookbook to this one first - GeoSciML 3.2 Encoding Cookbook for OneGeology 5 star WFS services
- The reader has some familiarity with setting up web servers and preferably Java servlet containers.
- The reader is able to install software on their machine and can follow the appropriate installation instructions for PostgreSQL, PostGIS and GeoServer documented on the websites for that software.

To set up a production OneGeology WFS Download service you will need server equipment to run your database and GeoServer. Estimating the level of hardware resources required to support a responsive service is a complex task depending on the amount of your data, its complexity and the demand that will be placed on your service by users. It is out of the scope of this document to give advice on these issues but you can find some assistance from the GeoServer web site and mailing list. To test setting up a service using the example in this guide a modern PC with Intel Core or equivalent processor and 4Gb RAM should certainly be adequate and you can probably get away with less.

The software required includes the PostgreSQL database with PostGIS spatial extensions, Java, in most cases a Java servlet container such as Apache Tomcat, and GeoServer itself. There are a lot of different versions and all these software packages are continuously updated. You may have conditions specific to your site which make particular versions preferable. For example, you may already have your data in a different database system such as Oracle Spatial rather than PostGIS. It isn't possible to cover all possible set ups here. We have run the example configuration on Windows 7, Windows Server 2008 OS's, with PostgreSQL v8.4 and v9.1 and PostGIS 1.5, Oracle Java 1.6 and 1.7 and GeoServer 2.4.5. Other versions will also be possible but you may encounter particular issues that require consulting the application documentation or mailing lists. Windows, Linux and Mac operating systems are all supported by the software used here although installation methods differ. PostgreSQL v8.4 will reach its end-of-life support in July 2014 so you should probably select a later version. GeoServer recommends the use of Oracle Java 1.6. See the GeoServer site for the latest recommendations. At the time of writing (March 2014) there were still some tests reported failing with Oracle Java 1.7 although we didn't encounter any problems in our example service with it. You should use GeoServer 2.4.5 or later as we have encountered bugs with the complex feature support in earlier versions.
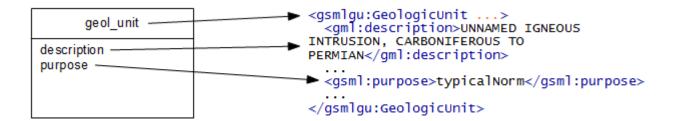
Overview

The text in this cookbook will assume you have your source data in tables in a relational database system. This is the most likely situation for data of any complexity but you should be able to adapt the details for other sources such as files etc.

You will need to study the GeoSciML data model. Having done this you will need to work out a mapping from your source data to appropriate parts of the GeoSciML model. See the GeoSciML 3.2 Encoding Cookbook for OneGeology 5 star WFS services. You will then have the task of configuring your software to generate XML conformant to your INSPIRE theme schema from your source data. The structure of your source data may be similar to or quite different from the INSPIRE model and we can't cover all possible situations but you will be able to make configuration changes in two places (i) in your underlying database and (ii) in the WFS software configuration mapping database contents to generated output. A number of constraints; administrative, performance, flexibility of the WFS software configuration and complexity will govern exactly how you do this but, as a general principle, it will usually be preferable to do as much in the database as possible to create source data matching INSPIRE output as closely as possible. GeoServer can do some quite powerful processing of the underlying data but this is often at the expense of performance and ability to query the data.
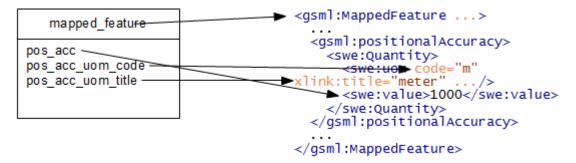
Normally you will have one main table for each feature type you are intending to serve. In a simple feature WFS each row will generate one feature and the values of the table columns will be the values contained in simple property values of the generated features (or spatial

data columns generating GML geometry property values). The diagram below shows a table `geol_unit` which is mapped to the feature type `gsmlgu:GeologicUnit`. Two example text columns `description` and `purpose` are mapped directly to simple feature properties `gml:description` and `gsml:purpose` with their text content being put directly inside the property elements.

With complex features such as in the INSPIRE data specifications there are a number of ways in which the configuration can be more complex.

A feature property could be 1-1 but its value a complex type with attributes or nested element content whose values come from the table columns. In the below diagram the columns of the `mapped_feature` table are directly placed inside property elements or attribute values within a complex nested property structure.

Note

often attributes like the units of measure may be the same for all rows in your table and you may want to hard code a value such as "meter" in the service configuration rather than repeat the same values in every row of your database table.

The property values may be polymorphic having different content structure depending on the particular data under consideration. A simple example would be nillable properties where the output structure changes when the property has nil values.

```
geol_event ──────   If process_uri and title have values
                    <gsmlga:GeologicEvent ...>
process_uri ──────     ...
process_title ─────     <gsmlga:eventProcess
                         xlink:href="
                    http://resource.geosciml.org/classifier/cgi/eventprocess/0041"
                          xlink:title="intrusion"/>
                        ...
                    </gsmlga:GeologicEvent>

                    If process_uri and title are null
                    <gsmlga:GeologicEvent ...>
                       ...
                        <gsmlga:eventProcess
                          xsi:nil="true"
                          xlink:title="intrusion"/>
                        ...
                    </gsmlga:GeologicEvent>
```

Other complexities include one to many, many to one or many to many relationships between features and their properties. In these cases you will normally have separate database tables for the features and their property values (which may be features in their own right).

Installation of Software

The software used here all has extensive documentation and support forums and mailing lists. Here we will just point you to the appropriate places to download the software and get installation instructions.

We assume:

- You have a basic familiarity with relational databases
- You are able to install applications such as PostGIS, Apache Tomcat, GeoServer etc. on a server with your chosen operating system using their project documentation.

Database

If you already have your data in a relational database system you will want to check the Working with Databases section of the GeoServer manual to see if your database is supported by GeoServer and follow the instructions for installation of any extensions that may be needed for this support. PostGIS support is built into the core GeoServer download.

If you want to try the example data or don't have a supported database system you should install PostGIS. The PostGIS Installation page contains instructions on how to do this for different operating systems. For the purposes of testing this cookbook setup we used the Enterprise DB Windows Installer which installs the base PostgreSQL database, has an

option to install the PostGIS extension into this and also includes the pgAdmin graphical database administration tool.

You may use a database on the same machine as GeoServer or you can have it on a separate machine which is accessible over a network from your GeoServer machine.

GeoServer

GeoServer has extensive documentation which you should refer to in addition to this cookbook. References will be made to relevant parts rather than repeating too much of what is already there. There is also a the geoserver-users mailing list.
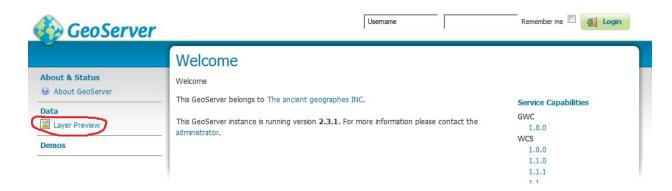
Assumptions

You will need to have Java installed on the machine on which you are going to install GeoServer. The GeoServer manual recommends Oracle JRE 6 or later but at the time of writing (March 2014) GeoServer was not fully tested with Oracle JRE 7. While testing the example for this document we used Oracle JRE 7 without apparent problems. Some people have successfully run it with non-Oracle versions of Java but these aren't supported and you may need to do extra work and be prepared to enter into technical discussions on the geoserver-users email list in order to get it working with these.

Ideally you will be familiar with deploying applications in a servlet container application such as Apache Tomcat or Jetty. If not, however, you can use one of the all-in-one installers to install a stand-alone version of GeoServer. Also, note that Tomcat and Jetty are the two most tested servlet containers used with GeoServer so, although you may well be able to use others, there may be extra work involved and you may need to get advice from the geoserver-users email list.

Basic Installation

- Download and install GeoServer following whichever one of the installation paths in the GeoServer manual suits your situation best. If you already have a servlet container application such as Apache Tomcat set up or you are familiar with how to set one up then you will probably wish to download the web archive (WAR) and deploy that in your servlet container. If you are not comfortable with configuring a servlet container then you will probably wish to use one of the installer programs.
- Run GeoServer at least once to check the installation has worked. If, for example, you are testing in a local instance of Tomcat on your own machine with default settings you should be able to visit http://localhost:8080/geoserver and click on the *Layer Preview* link to check that the example services shipped with GeoServer work

properly. You will have to modify the preceding URL appropriately if you have deployed GeoServer on a different machine or port.

App-schema Plugin Installation

- Download and install the app-schema plugin `geoserver-*-app-schema-plugin.zip` following the application schema installation instructions. Note the WFS service settings.
- Check you can restart GeoServer without any errors being thrown.

INSPIRE Plugin Installation

If you are providing an INSPIRE download service you will need to provide the extra INSPIRE mandated metadata in the WFS GetCapabilities response. This is optional for OneGeology.

- Download and install the INSPIRE plugin `geoserver-*-inspire-plugin.zip` following the INSPIRE plugin installation instructions.
- Check that the WFS 1.1.0 and 2.0.0 GetCapabilities responses contain the `inspire_dls:ExtendedCapabilities` section.

Example Data and Configuration

This tutorial chapter will show you how to set up a OneGeology compliant GeoSciML v3.2 WFS covering at least the encoding guidance in the GeoSciML 3.2 Encoding Cookbook for OneGeology 5 star WFS services. Version 1.0 using the Open Source GeoServer WFS and PostGIS spatial database with an example source dataset. It enables you to get some initial experience setting up a service. Your own services may be set up by customising it or you may use it to get some understanding of what is involved when setting up your own service.

The tutorial assumes you are familiar with representing complex features in GML applications.

Loading example PostGIS data

It is assumed that you have installed the PostGIS software and that you have a spatially enabled database (the default installation will create one called postgis). The following will all be working within this spatially enabled database.

Download the database dump file from ftp://ftp.bgs.ac.uk/pubload/OneGeology/INSPIREWFS/gbr_bgs_625k.backup.

Create a separate schema for the example data. The schema name is used in the commands below so, if you change it from `gbr_bgs_625k` you will need to change these accordingly

```
CREATE SCHEMA gbr_bgs_625k AUTHORIZATION postgres;
```

Import the data from the database dump file downloaded above. If you have installed the pgAdmin graphical administration tool you can use the menu option *Tools ‣ Restore...*. If you are using the command line you can use the command **pg_restore.exe –host localhost – port 5432 –username "postgres" –dbname "postgis" –no-password –no-owner –no-privileges –no-tablespaces –schema gbr_bgs_625k –verbose gbr_bgs_625k.backup** (assuming you are using the default 'postgis' named database. You should create a database user with read-only access to these tables for the WFS software to use when accessing them. With more recent versions of PostgreSQL you can use the shorter syntax to grant access to all tables in the gbr_bgs_625k schema, with older versions you will have to grant access to each table individually

```
CREATE ROLE ows_reader LOGIN PASSWORD 'your_password'
 VALID UNTIL 'infinity';
COMMENT ON ROLE ows_reader
 IS 'A role with read only access to data used in web services.';
grant usage on schema gbr_bgs_625k to ows_reader;
grant select on table geometry_columns to ows_reader;
grant select on table spatial_ref_sys to ows_reader;
```

Short syntax for versions of PostgreSQL (works at least as far back as v9.1)

```
grant select on all tables in schema gbr_bgs_625k to ows_reader;
```

Separate grants for each table for older versions of PostgreSQL (known to be required for v8.4)

```
grant select on table gbr_bgs_625k.uk_625k_mapped_feature to ows_reader;
grant select on table gbr_bgs_625k.uk_625k_geol_unit to ows_reader;
```

```
grant select on table gbr_bgs_625k.uk_625k_geol_unit_comp_part to
ows_reader;
```

If you have problems with the above steps which are difficult to resolve you may find that setting `log_statement=all` in `postgresql.conf`, reloading the server and then monitoring the log file is helpful for debugging them.

Configuring GeoServer WFS

Download the example configuration files from ftp://ftp.bgs.ac.uk/pubload/OneGeology/gbr_bgs_625k_wfs/data_625k_cgi.zip and expand it to a spare location on your server. Copy the files from this expanded directory to the matching locations in your GeoServer data directory, overwriting any files that already exist there. The main configuration files are inside the `workspaces` directory, these shouldn't exist in your installation except for the `default.xml` file which simply defines the example gsml:MappedFeature to be the default. The files (`global.xml`, `wfs.xml`,...) at the top level of the directory contain service metadata relevant to the example data set. When you set up your own data you will use the web GUI to change these values. The contents of the `demo` directory are some example requests which are documented below.

Copy the file ftp://ftp.bgs.ac.uk/pubload/OneGeology/INSPIREWFS/app-schema.properties to `WEB-INF/classes/app-schema.properties` and edit the database connection parameters appropriately for wherever your installation of PostgreSQL is running. If you want to use a JNDI data connection configured in your servlet container then you will also need to edit the appropriate places in the `datastore.xml` files described in a subsequent section. So it will be easier for initial testing just to enter the host, database, user and password parameters.

Perform any configuration required by your servlet container, and then start the servlet.

One configuration item you may need to change is to increase the memory available for Java. The method depends on how you have installed GeoServer but if you get `java.lang.OutOfMemoryError: Java heap space` errors with the request below you will need to increase the memory with a directive such as `-Xmx256M`. The details of tuning memory and other options of the Java Virtual Machine are complex and not dealt with in this cookbook. Some information is in the GeoServer User Manual under the Running in a Production Environment section.

If you have used the Windows Installer you can apply this by editing the file `C:\Program Files (x86)\GeoServer 2.4.5\wrapper\wrapper.conf` (The exact file location will depend on where you installed GeoServer and which version you are using.) Find the

line `wrapper.java.maxmemory=128` and increase the value 128 (or whatever it happens to be) to something like 256.

If you are running in Apache Tomcat on Windows you can use the "Configure Tomcat" program that the Tomcat Windows installer provides. In the "Java" tab you can put a maximum memory value such as 256 (MB) in the Maximum Memory pool field.
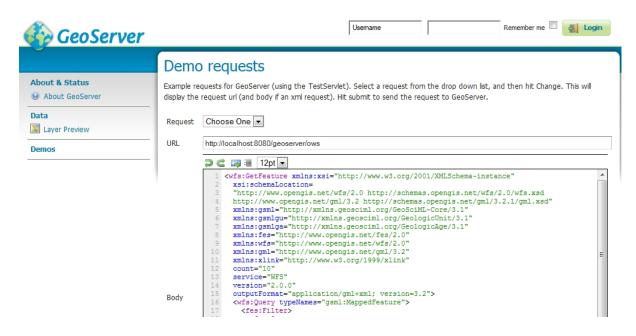
- The first time GeoServer starts with the tutorial configuration, it will download all the schema (XSD) files it needs and store them in the `app-schema-cache` folder in the data directory. **You must be connected to the internet for this to work.**

Complex Feature Test requests

When GeoServer is running, test app-schema WFS in a web browser. You can query the six feature types using these links. (Change `localhost:8080` in the examples below if you have deployed it at a different location.):

- http://localhost:8080/geoserver/wfs?request=GetFeature&version=2.0.0& typeName=gsml:MappedFeature&count=25
- http://localhost:8080/geoserver/wfs?request=GetFeature&version=2.0.0& typeName=gsmlgu:GeologicUnit&count=25

You can also obtain WFS responses by using the Demo requests page in the GeoServer web interface. (Note that the web interface does not yet support app-schema store for layer administration.) You can select some of the example age and lithology queries from the request drop-down list or put `http://localhost:8080/geoserver/ows` into the service URL section and try pasting in your own queries. The examples are reproduced below.

A query for mapped features showing outcrops of geologic units of a particular age (`WFS_getFeature1GAge.xml`)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
  http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd
  http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:gsmlgu="http://xmlns.geosciml.org/GeologicUnit/3.2"
 xmlns:gsmlga="http://xmlns.geosciml.org/GeologicAge/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 xmlns:xlink="http://www.w3.org/1999/xlink" count="10" service="WFS"
 version="2.0.0" outputFormat="application/gml+xml; version=3.2">
 <wfs:Query typeNames="gsml:MappedFeature">
  <fes:Filter>
   <fes:Or>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsml:relatedFeature/gsmlga:GeologicHistory/gsml:relatedFeature/gsmlga:GeologicEvent/gsmlga:youngerNamedAge/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/ics/ischart/LowerDevonian</fes:Literal>
    </fes:PropertyIsEqualTo>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsml:relatedFeature/gsmlga:GeologicHistory/gsml:relatedFeature/gsmlga:GeologicEvent/gsmlga:youngerNamedAge/@xlink:href</fes:ValueReference>
```

```
<fes:Literal>http://resource.geosciml.org/classifier/ics/ischart/Emsian</fe
s:Literal>
    </fes:PropertyIsEqualTo>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsml:relatedFeat
ure/gsmlga:GeologicHistory/gsml:relatedFeature/gsmlga:GeologicEvent/gsmlga:
youngerNamedAge/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/ics/ischart/Lochkovian
</fes:Literal>
    </fes:PropertyIsEqualTo>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsml:relatedFeat
ure/gsmlga:GeologicHistory/gsml:relatedFeature/gsmlga:GeologicEvent/gsmlga:
youngerNamedAge/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/ics/ischart/Pragian</f
es:Literal>
    </fes:PropertyIsEqualTo>
  </fes:Or>
 </fes:Filter>
 </wfs:Query>
</wfs:GetFeature>
```

A OneGeology query for mapped features showing outcrops of geological units with particular lithologies (`WFS_getFeature1GLith.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
  "http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd
  http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:gsmlgu="http://xmlns.geosciml.org/GeologicUnit/3.2"
 xmlns:gsmlga="http://xmlns.geosciml.org/GeologicAge/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 xmlns:xlink="http://www.w3.org/1999/xlink" count="10" service="WFS"
 version="2.0.0" outputFormat="application/gml+xml; version=3.2">
 <wfs:Query typeNames="gsml:MappedFeature">
  <fes:Filter>
   <fes:Or>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsmlgu:compositi
on/gsmlgu:CompositionPart/gsmlgu:material/gsmlem:RockMaterial/gsmlem:lithol
ogy/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/cgi/lithology/limeston
e</fes:Literal>
```

```
    </fes:PropertyIsEqualTo>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsmlgu:compositi
on/gsmlgu:CompositionPart/gsmlgu:material/gsmlem:RockMaterial/gsmlem:lithol
ogy/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/cgi/lithology/chalk</f
es:Literal>
    </fes:PropertyIsEqualTo>
    <fes:PropertyIsEqualTo>
<fes:ValueReference>gsml:specification/gsmlgu:GeologicUnit/gsmlgu:compositi
on/gsmlgu:CompositionPart/gsmlgu:material/gsmlem:RockMaterial/gsmlem:lithol
ogy/@xlink:href</fes:ValueReference>
<fes:Literal>http://resource.geosciml.org/classifier/cgi/lithology/traverti
ne</fes:Literal>
    </fes:PropertyIsEqualTo>
   </fes:Or>
  </fes:Filter>
 </wfs:Query>
</wfs:GetFeature>
```

A bounding box query to retrieve mapped features with shapes that overlap the specified bounding box (`WFS_getFeatureBBOX.xml`)

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd
  http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns="http://www.opengis.net/wfs/2.0"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:gml="http://www.opengis.net/gml/3.2" version="2.0.0" service="WFS"
 count="100">
 <Query typeNames="gsml:MappedFeature"
srsName="urn:ogc:def:crs:EPSG::4326">
  <fes:Filter>
   <fes:BBOX>
    <fes:ValueReference>gsml:shape</fes:ValueReference>
    <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">
     <gml:lowerCorner>56.08643859340388 -
4.0004826736994445</gml:lowerCorner>
     <gml:upperCorner>56.165510496146474 -
3.8381055732299343</gml:upperCorner>
    </gml:Envelope>
   </fes:BBOX>
  </fes:Filter>
 </Query>
```

```
</GetFeature>
```

Note

App-schema cannot be configured using the web interface, you will need to edit the configuration files directly. You will see the configured workspaces and stores appear in the web interface but not the layers (features). The properties that can be edited in the web interface are very limited.

INSPIRE Extended Capabilities

If you are providing an INSPIRE download service you will need to provide the extra INSPIRE mandated metadata in the WFS GetCapabilities response. All INSPIRE specific requirements are optional for OneGeology use but will work as OneGeology services.

As described in the plugin documentation you should find a section in the WFS service settings of the administration interface where you can choose a language, enter a service metadata URL and type and add one or more spatial dataset identifiers. For guidance on what to enter in these settings see the Technical Guidance for the implementation of INSPIRE Download Services.

Check that the GetCapabilities responses contain your edited values.

Note

The plugin GUI only allows you to select one of two values for the Service Metadata Type: "Online ISO 19139 ServiceMetadata document" sets the MIME type to `application/vnd.iso.19139+xml`, "CSW GetRecord by ID request" sets the MIME type to `application/vnd.ogc.csw.GetRecordByIdResponse_xml`. If neither of these correspond to the actual MIME type of your metadata resource you will need to work around this by manually editing the wfs.xml file inside the GeoServer data directory as documented in issue GEOS-5157 . On the other hand it isn't clear whether any client would actually use this information.

For example, entering the values shown in the screenshot below would result in a GetCapabilities response with the ExtendedCapabilities section shown below it.

INSPIRE

Language

eng ▾

Service Metadata URL

http://metadata.bgs.ac.uk/geonetwork/srv/en/csw?SE

Service Metadata Type

CSW GetRecord by ID request ▾

Spatial Dataset Identifiers

Code	Namespace	
9df8df51-6342-37a8-e044-0003ba9b0d98		Remove

Add identifier

```xml
<ows:ExtendedCapabilities>
 <inspire_dls:ExtendedCapabilities>
  <inspire_common:MetadataUrl
   xsi:type="inspire_common:resourceLocatorType">
   <inspire_common:URL>
http://metadata.bgs.ac.uk/geonetwork/srv/en/csw?SERVICE=CSW&amp;REQUEST=Get
RecordById&amp;elementSetName=full&amp;OutputSchema=http://www.isotc211.org
/2005/gmd&amp;ID=7822e848-822d-45a5-8584-56d352fd2170&amp;
   </inspire_common:URL>
   <inspire_common:MediaType>
    application/vnd.ogc.csw.GetRecordByIdResponse_xml
   </inspire_common:MediaType>
  </inspire_common:MetadataUrl>
  <inspire_common:SupportedLanguages
   xsi:type="inspire_common:supportedLanguagesType">
   <inspire_common:DefaultLanguage>
    <inspire_common:Language>eng</inspire_common:Language>
   </inspire_common:DefaultLanguage>
  </inspire_common:SupportedLanguages>
  <inspire_common:ResponseLanguage>
   <inspire_common:Language>eng</inspire_common:Language>
  </inspire_common:ResponseLanguage>
  <inspire_dls:SpatialDataSetIdentifier>
   <inspire_common:Code>
    9df8df51-6342-37a8-e044-0003ba9b0d98
   </inspire_common:Code>
  </inspire_dls:SpatialDataSetIdentifier>
 </inspire_dls:ExtendedCapabilities>
</ows:ExtendedCapabilities>
```

INSPIRE Pre-defined Dataset Download

If you decide that you are going to provide an INSPIRE pre-defined dataset download service direct from your WFS rather than pre-generating the full datasets and just providing links to the download through ATOM then you can do this by creating a Stored Query such as the one below. The example data directory includes the CreateStoredQuery command in the Demos examples.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:CreateStoredQuery
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
  "http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:fes="http://www.opengis.org/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 service="WFS"
 version="2.0.0">
 <wfs:StoredQueryDefinition
  id='http://inspire.ec.europa.eu/operation/download/GetSpatialDataSet'>
  <wfs:Parameter name='CRS' type='xsd:string'/>
  <wfs:Parameter name='DataSetIdCode' type='xsd:string'/>
  <wfs:Parameter name='DataSetIdNamespace' type='xsd:string'/>
  <wfs:Parameter name='Language' type='xsd:string'/>
  <wfs:Parameter name="count" type="xsd:integer"></wfs:Parameter>
  <wfs:QueryExpressionText
   returnFeatureTypes='gsml:MappedFeature'
   language='urn:ogc:def:queryLanguage:OGC-WFS::WFS_QueryExpression'
   isPrivate='false'>
   <wfs:Query typeNames='gsml:MappedFeature' srsName="${CRS}">
   </wfs:Query>
  </wfs:QueryExpressionText>
 </wfs:StoredQueryDefinition>
</wfs:CreateStoredQuery>
```

This can then be invoked with a request like:

http://localhost:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetFeature& storedquery_id=http://inspire.ec.europa.eu/operation/download/GetSpatialDataSet& DataSetIdCode=13603180&DataSetIdNamespace=http://data.bgs.ac.uk/id/dataHolding/& CRS=urn:ogc:def:crs:EPSG::4326&Language=eng&count=20&

Complex Feature Configuration

This part of the cookbook describes selected parts of the configuration files from the example set. This should give you a good overview of the configuration files needed and what the different parts do. They produce valid INSPIRE Annex II Geology theme (GeoSciML v3.2) features and so, hopefully will provide a good starting point to adapt for your own services. However, the mapping from your source data to your INSPIRE theme model may require some features not used in the example. For these cases you should refer to the Working with Application Schemas section of the GeoServer manual which contains comprehensive documentation on the different kinds of mapping from source to output XML that are possible. (It uses GeoSciML v2 based examples.)

Because a single `gsmlgu:GeologicUnit` can be observed at several distinct locations on the Earth's surface, several `gsml:MappedFeature` features may point via their gsml:specification property to the same gsmlgu:GeologicUnit.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection>
 <!-- ... -->
 <wfs:member>
  <gsml:MappedFeature gml:id="bgsn_digmap20111213000014089_625k">
   <gml:boundedBy>
    <gml:Envelope srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
     <gml:lowerCorner>276992.0639999996 693085.1679999997</gml:lowerCorner>
     <gml:upperCorner>277851.8129999998 694098.9850000012</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <!-- ... -->
   <gsml:shape>
    <gml:MultiSurface gml:id="gsml.shape.bgsn_digmap20111213000014089_625k"
     srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
     <gml:surfaceMember>
      <gml:Polygon gml:id="gsml.shape.bgsn_digmap20111213000014089_625k.1"
       srsDimension="2">
       <gml:exterior>
        <gml:LinearRing>
         <gml:posList>277851.8129999998 693998.524000001 277823.5400000005
          693969.0079999991 277654.64600000007 693696.0569999999
          277576.6949999996 693449.1019999995 277550.7119999993
          693280.1319999992 277537.7199999998 693085.1679999997
          276992.0639999996 693202.1460000003 277018.0469999999
          693566.081000001 277134.9740000011 693839.0320000002
          277251.8999999991 694098.9850000012 277851.8129999998
          693998.524000001</gml:posList>
        </gml:LinearRing>
       </gml:exterior>
```

```
        </gml:Polygon>
      </gml:surfaceMember>
    </gml:MultiSurface>
   </gsml:shape>
   <gsml:specification>
    <gsmlgu:GeologicUnit gml:id="SYG-CYCS">
     <gml:description>STRATHCLYDE GROUP - SEDIMENTARY ROCK CYCLES,
      STRATHCLYDE GROUP TYPE</gml:description>
     <!-- ... -->
    </gsmlgu:GeologicUnit>
   </gsml:specification>
   <gsml:metadata xsi:nil="true" gco:nilReason="missing"/>
  </gsml:MappedFeature>
 </wfs:member>
 <wfs:member>
  <gsml:MappedFeature gml:id="bgsn_digmap20111213000015084_625k">
   <gml:boundedBy>
    <gml:Envelope srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
     <gml:lowerCorner>280207.97299999953
694960.4289999997</gml:lowerCorner>
     <gml:upperCorner>281182.8969999998 696298.8440000011</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <!-- ... -->
   <gsml:shape>
    <!-- ... -->
   </gsml:shape>
   <gsml:specification>
    <gsmlgu:GeologicUnit gml:id="CKN-CYCC">
     <gml:description>CLACKMANNAN GROUP - SEDIMENTARY ROCK CYCLES,
      CLACKMANNAN GROUP TYPE</gml:description>
     <!-- ... -->
    </gsmlgu:GeologicUnit>
   </gsml:specification>
   <gsml:metadata xsi:nil="true" gco:nilReason="missing"/>
  </gsml:MappedFeature>
 </wfs:member>
 <wfs:member>
  <gsml:MappedFeature gml:id="bgsn_digmap20111213000016262_625k">
   <gml:boundedBy>
    <gml:Envelope srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
     <gml:lowerCorner>273924.59559999984
687632.7340000008</gml:lowerCorner>
     <gml:upperCorner>276292.57699999935
690639.8490000011</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <!-- ... -->
   <gsml:shape>
    <!-- ... -->
   </gsml:shape>
```

```
    <gsml:specification xlink:href="#SYG-CYCS"/>
    <gsml:metadata xsi:nil="true" gco:nilReason="missing"/>
   </gsml:MappedFeature>
 </wfs:member>
 <wfs:member>
   <gsml:MappedFeature gml:id="bgsn_digmap20111213000016481_625k">
    <gml:boundedBy>
     <gml:Envelope srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
      <gml:lowerCorner>275822.8000000001 690776.9270000004</gml:lowerCorner>
      <gml:upperCorner>277358.40499999985
693332.1230000001</gml:upperCorner>
     </gml:Envelope>
    </gml:boundedBy>
     <!-- ... -->
    <gsml:shape>
     <!-- ... -->
    </gsml:shape>
    <gsml:specification xlink:href="#SYG-CYCS"/>
    <gsml:metadata xsi:nil="true" gco:nilReason="missing"/>
   </gsml:MappedFeature>
 </wfs:member>
 <wfs:member>
   <gsml:MappedFeature gml:id="bgsn_digmap20111213000023007_625k">
    <gml:boundedBy>
     <gml:Envelope srsDimension="2" srsName="urn:ogc:def:crs:EPSG::27700">
      <gml:lowerCorner>247904.646999999 664467.5469999999</gml:lowerCorner>
      <gml:upperCorner>289537.88300000026
697616.4020000009</gml:upperCorner>
     </gml:Envelope>
    </gml:boundedBy>
     <!-- ... -->
    <gsml:shape>
     <!-- ... -->
    </gsml:shape>
    <gsml:specification xlink:href="#CKN-CYCC"/>
    <gsml:metadata xsi:nil="true" gco:nilReason="missing"/>
   </gsml:MappedFeature>
 </wfs:member>
</wfs:FeatureCollection>
```

app-schema.properties

This file (in the `WEB-INF/classes` directory) is not strictly required but is very useful for storing certain configuration parameters that will be re-used in different parts of the other configuration files and for storing configuration parameters like database usernames and passwords outside of the GeoServer data directory so that the latter can be copied freely. The Property Interpolation section of the GeoServer manual contains more information on how properties can be set and used in other parts of the configuration files. In the reference

configuration files this contains database connection parameters, some commonly used URI values and the setting to turn on joining. In fact with current versions of GeoServer (certainly pre-dating v2.4.5) joining is turned on by default and is the recommended setting. There may be some limited situations as described in the joining documentation, however, when you need to switch this off.

Data directory

The example configuration files contain a complete GeoServer data directory so that you can get a working service to try out up and running as quickly as possible. The parts relevant for configuration of a complex feature WFS are contained in the `workspaces` directory described in the next section. The other parts contain items like service metadata, security etc. and can mostly be set up using the web interface. Thus, when you come to set up your own service, you will probably start with the default GeoServer data directory, configure service metadata etc. in the web interface, and copy the complex feature configuration files from the reference `data/workspaces` directory to your own data directory for editing there.

Workspace layout

The files for configuring complex feature output are contained in the `data/workspaces` directory. Inside this directory there is a sub-directory for each namespace of features you will be serving and other namespaces that these features may use somewhere in their content. In the example this includes:

```
workspaces
├── gco
├── gmd
├── gml
├── gsml
├── gsmlem
├── gsmlga
├── gsmlgs
├── gsmlgu
├── gsmlu
├── swe
└── xlink
```

These cover all the namespaces used by features in the example data set so, if you are producing a Geology theme service, it is likely that you will use the same ones. However, for other themes you will need to create similar directories for the namespaces used in those themes and, even for the geology theme, you may need to add others if you use additional GeoSciML packages such as boreholes.

The example configuration defines 2 feature types to be served by the WFS. Their configurations are stored in data store sub-directories of the appropriate namespace

directory and are named according to the pattern `prefix_Feature` for a feature `prefix::Feature`:

```
workspaces
├── gco
├── gmd
├── gml
├── gsml
│   └── gsml_MappedFeature
├── gsmlem
├── gsmlga
├── gsmlgs
├── gsmlgu
│   └── gsmlgu_GeologicUnit
├── gsmlu
├── swe
└── xlink
```

Each of the data store directories contains files similar to the following example for `gsml::MappedFeature`:

```
gsml_MappedFeature
├── AppSchemaDataAccess.xsd
├── datastore.xml
├── gsml_MappedFeature
│   └── featuretype.xml
└── gsml_MappedFeature.xml
```

The `AppSchemaDataAccess.xsd` file isn't used for the configuration, it is just provided as a convenience when you are editing a mapping file such as `gsml_MappedFeature.xml` to allow a validating XML editor to give you hints that you have the structure of the file correct. The following sections will describe the `datastore.xml` file which creates an application schema data store and specifies the mapping file described in the section after which contains the substantive portion of the configuration mapping source data fields to output in the complex feature types.

datastore.xml

Each data store configuration file `datastore.xml` specifies the location of a mapping file and triggers its loading as an app-schema data source. This file should not be confused with the source data store, which is specified inside the mapping file.

For `gsml:MappedFeature` the file is `workspaces/gsml/gsml_MappedFeature/datastore.xml`

```xml
<dataStore>
 <id>gsml_MappedFeature_datastore</id>
```

```
<name>gsml_MappedFeature</name>
<enabled>true</enabled>
<workspace>
 <id>gsml_workspace</id>
</workspace>
<connectionParameters>
 <entry key="namespace">http://xmlns.geosciml.org/GeoSciML-
Core/3.2</entry>
 <entry
key="url">file:workspaces/gsml/gsml_MappedFeature/gsml_MappedFeature.xml</e
ntry>
 <entry key="dbtype">app-schema</entry>
</connectionParameters>
</dataStore>
```

Note

Ensure that there is no whitespace inside an `entry` element.

For other feature types the pattern is the same where you replace the names and ids appropriately and change the namespace if necessary. The url entry is a file: URI pointing to the location of the mapping file relative to the GeoServer data directory. The dbtype entry will always be app-schema to define complex feature types.

Mapping files

Configuration of app-schema feature types is performed in mapping files e.g.

- `workspaces/gsmlgu/gsmlgu_GeologicUnit/gsmlgu_GeologicUnit.xml`
- `workspaces/gsml/gsml_MappedFeature/gsml_MappedFeature.xml`

Namespaces

Each mapping file contains namespace prefix definitions. The below extract is from `gsml_MappedFeature.xml`

```
<namespaces>
 <Namespace>
  <prefix>gml</prefix>
  <uri>http://www.opengis.net/gml/3.2</uri>
 </Namespace>
 <Namespace>
  <prefix>gsml</prefix>
  <uri>http://xmlns.geosciml.org/GeoSciML-Core/3.2</uri>
 </Namespace>
 <Namespace>
```

```
   <prefix>gsmlgu</prefix>
   <uri>http://xmlns.geosciml.org/GeologicUnit/3.2</uri>
 </Namespace>
 <Namespace>
   <prefix>gsmlgs</prefix>
   <uri>http://xmlns.geosciml.org/GeologicStructure/3.2</uri>
 </Namespace>
 <Namespace>
   <prefix>swe</prefix>
   <uri>http://www.opengis.net/swe/2.0</uri>
 </Namespace>
 <Namespace>
   <prefix>gmd</prefix>
   <uri>http://www.isotc211.org/2005/gmd</uri>
 </Namespace>
 <Namespace>
   <prefix>gco</prefix>
   <uri>http://www.isotc211.org/2005/gco</uri>
 </Namespace>
 <Namespace>
   <prefix>xlink</prefix>
   <uri>http://www.w3.org/1999/xlink</uri>
 </Namespace>
 <Namespace>
   <prefix>xsi</prefix>
   <uri>http://www.w3.org/2001/XMLSchema-instance</uri>
 </Namespace>
</namespaces>
```

Only those namespace prefixes used in the mapping file need to be declared although you may find it easier just to put all the namespaces used by GeoSciML in all of them.

Source data store

The data for this tutorial is contained in the PostGIS database set up in the previous section.

For this example, each feature type uses an identical source data store configuration. The `Parameter` elements contain values for various database connection parameters. Here we are using property interpolation so that these don't have to get changed in each mapping file if they change and so that the configuration files can be shared without exposing password information. The values of the interpolated variables (which have the form `${name}`) should be defined in the `WEB-INF/classes/app-schema.properties` file. An example which you can use as a template to fill in with your own values is at ftp://ftp.bgs.ac.uk/pubload/OneGeology/INSPIREWFS/app-schema.properties.

```
<sourceDataStores>
 <DataStore>
```

```
  <id>datastore</id>
  <parameters>
   <Parameter>
    <name>dbtype</name><value>${cgi.dbtype}</value>
   </Parameter>
   <!--
   <Parameter>
    <name>jndiReferenceName</name><value>${cgi.jndi}</value>
   </Parameter>
   -->
   <Parameter>
    <name>host</name><value>${cgi.host}</value>
   </Parameter>
   <Parameter>
    <name>port</name><value>${cgi.port}</value>
   </Parameter>
   <Parameter>
    <name>database</name><value>${cgi.database}</value>
   </Parameter>
   <Parameter>
    <name>user</name><value>${cgi.user}</value>
   </Parameter>
   <Parameter>
    <name>passwd</name><value>${cgi.passwd}</value>
   </Parameter>
   <Parameter>
    <name>schema</name><value>${cgi.schema}</value>
   </Parameter>
   <Parameter>
    <name>Expose primary keys</name><value>true</value>
   </Parameter>
  </parameters>
 </DataStore>
</sourceDataStores>
```

See   http://docs.geoserver.org/stable/en/user/data/app-schema/data-stores.html   for   a
description of how to use other types of data store.

Target types

The XML Schemas which are required to define a feature type and its properties are
specified in the `targetTypes` section. The type of the output feature is defined in
`targetElement` in the `typeMapping` section. The below example is from
`gsml_MappedFeature.xml`

```
<targetTypes>
 <FeatureType>
  <schemaUri>
```

```
   http://schemas.geosciml.org/geologicunit/3.2/geologicUnit.xsd
  </schemaUri>
 </FeatureType>
 <FeatureType>
  <schemaUri>
   http://schemas.geosciml.org/geologicstructure/3.2/geologicStructure.xsd
  </schemaUri>
 </FeatureType>
</targetTypes>
```

In this case the schema is published, but because the OASIS XML Catalog is used for schema resolution, a private or modified schema in the catalog can be used if desired.

Mappings

The `typeMappings` element begins with configuration elements. From `gsml_MappedFeature.xml`

```
<typeMappings>
 <FeatureTypeMapping>
  <sourceDataStore>datastore</sourceDataStore>
  <sourceType>uk_625k_mapped_feature</sourceType>
  <targetElement>gsml:MappedFeature</targetElement>
```

- The mapping starts with `sourceDataStore`, which gives the arbitrary identifier used above to name the source of the input data in the `sourceDataStores` section.
- `sourceType` gives the name of the source simple feature type. In this case it is the simple feature type `gsml_mappedfeature`, sourced from the table of the same name in the PostGIS database set up in the previous chapter.
- When working with databases `sourceType` is the name of a table or view. Database identifiers must be lowercase for PostGIS or uppercase for Oracle Spatial.
- `targetElement` is the name of the output complex feature type.

gml:id mapping

The first mapping sets the `gml:id` to be the feature id specified in the source property file

```
<AttributeMapping>
 <targetAttribute>gsml:MappedFeature</targetAttribute>
 <idExpression><OCQL>uuid</OCQL></idExpression>
</AttributeMapping>
```

- `targetAttribute` is the XPath to the element for which the mapping applies, in this case, the top-level feature type.
- `idExpression` is a special form that can only be used to set the `gml:id` on a feature. In the tutorial database we have a column `uuid` suitable for use as a `gml:id` attribute value. If your database doesn't have such a column you may be able to use the special function `getId()` here instead. This will synthesise an id from the table or view name, a dot ".", and the primary key of the table. If this is not desirable, any other field or CQL expression can be used, if it evaluates to an NCName.

The above would generate output like

```
<gsml:MappedFeature gml:id="bgsn_digmap20111213000014089_625k">
...
```

Ordinary mapping

Most mappings consist of a target and source. Here is a simple property with a text value from `gsmlgu_GeologicUnit.xml`

```
<AttributeMapping>
 <targetAttribute>gml:description</targetAttribute>
 <sourceExpression>
  <OCQL>description</OCQL>
 </sourceExpression>
</AttributeMapping>
```

- In this case, the value of `gml:description` is just the value of the `description` column in the `uk_625k_geol_unit` table.
- For a database, the field name is the name of the column (the table/view is set in `sourceType` above). Database identifiers must be lowercase for PostGIS or uppercase for Oracle Spatial.
- CQL expressions can be used to calculate content. Use caution because queries on CQL-calculated values prevent the construction of efficient SQL queries.
- Source expressions can be CQL literals, which are single-quoted.

The above would generate output like

```
<gml:description>STRATHCLYDE GROUP - SEDIMENTARY ROCK CYCLES,
 STRATHCLYDE GROUP TYPE</gml:description>
```

Client properties

In addition to the element content, a mapping can set values of XML attributes on property elements. These are called "client properties" in GeoServer terminology. Here is one from `gsmlgu_CompositionPart.xml`

```
<AttributeMapping>

<targetAttribute>gsmlgu:material/gsmlem:RockMaterial/gsmlem:lithology</targetAttribute>

<ClientProperty><name>xlink:href</name><value>cgi_lithology_uri</value></ClientProperty>

<ClientProperty><name>xlink:title</name><value>cgi_lithology_label</value></ClientProperty>
</AttributeMapping>
```

- This mapping leaves the content of the `gsmlem:lithology` element empty but sets an `xlink:href` attribute to the value of the `cgi_lithology_uri` column and an `xlink:title` attribute to the value of the `cgi_lithology_label` column. Note also that in this case the property attributes being set are nested two levels below the parent `gsmlgu:CompositionPart`.
- As can be seen from this example, multiple `ClientProperty` mappings can be set.

This would generate output like

```
<gsmlem:lithology
 xlink:href=
  "http://resource.geosciml.org/classifier/cgi/lithology/clastic_mudstone"
 xlink:title="Mudstone"/>
```

Feature chaining

In feature chaining, one feature type is used as a property of an enclosing feature type, by value or by reference. The following examples show the parts of the `gsml_MappedFeature.xml` and `gsmlgu_GeologicUnit.xml` mapping files that put the gsmlgu:GeologicFeature elements inside the gsml:specification properties of gsml:MappedFeature elements.

In `gsml_MappedFeature.xml`

```
<AttributeMapping>
 <targetAttribute>gsml:specification</targetAttribute>
 <sourceExpression>
  <OCQL>lex_rcs</OCQL>
  <linkElement>gsmlgu:GeologicUnit</linkElement>
  <linkField>FEATURE_LINK</linkField>
 </sourceExpression>
</AttributeMapping>
```

In `gsmlgu_GeologicUnit.xml`

```
<AttributeMapping>
 <targetAttribute>FEATURE_LINK</targetAttribute>
 <sourceExpression><OCQL>lex_rcs</OCQL></sourceExpression>
</AttributeMapping>
```

- In this case from the mapping for `gsml:MappedFeature`, we specify a mapping for its `gsml:specification` property.
- The `linkElement` specifies which feature (or other complex type) should be used as the value of the property.
- The `link_specification` field of the source `gsml_MappedFeature` simple feature is use as the "foreign key", which maps to the special `FEATURE_LINK` field in each `gsml:GeologicFeature`.
- The mapping of the special `FEATURE_LINK` attribute in `gsmlgu_GeologicFeature.xml` to the foreign key field of the underlying table means that every `gsmlgu:GeologicFeature` with `lex_rcs` equal to the `lex_rcs` of the `gsml:MappedFeature` under construction is included as a `gsml:specification` property of the `gsml:MappedFeature` (by value).

Feature chaining has been used to construct the property `gsml:specification` of `gsml:MappedFeature`. This property is a `gsmlgu:GeologicUnit`. The WFS response for `gsml:MappedFeature` combines the output of both feature types into a single response. The first `gsml:MappedFeature` has a `gsml:specification` property value of the `gsmlgu:GeologicUnit` with the same `lex_rcs` code. The next time a mapped feature with the same `lex_rcs` code appears, rather than including the whole geologic unit inline again the property has an xlink:href attribute pointing to the first occurrence. The relationships between the feature instances are data driven.

Portal Age and Lithology Query

The OneGeology portal has a graphical interface to allow a user to select geological ages or lithologies from a hierarchical list and query a given WFS so that it returns all features with the selected property values within the current viewing window. In order to work with this your WFS must include the features and properties and dictionaries required by the query and described in the GeoSciML 3.2 Encoding Cookbook for OneGeology 5 star WFS services. Version 1.0. This chapter gives you some example queries which your WFS should support and shows how these are created in the OneGeology portal.

When you have set up a WFS for OneGeology you will be able to ask for it to be registered against one or more WMS layers that portray the data returned by it (possibly styled differently according to different property values). If you then select one of those registered WMS layers in the portal and select the *Active Layer Properties* you will see a dialog box like below:

If you expand the layer description by clicking on the down arrow at the right you should see buttons for GeoSciML WFS URL and Thematic Analysis at the bottom.
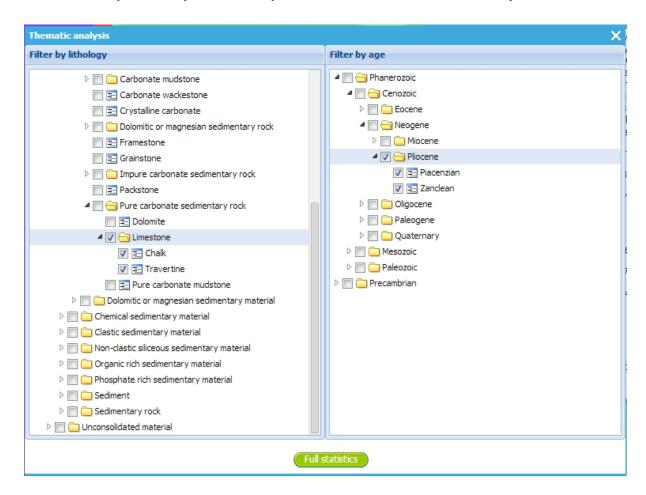

Clicking on the Thematic Analysis button will bring a choice of Full Statistics or Lithology Statistics
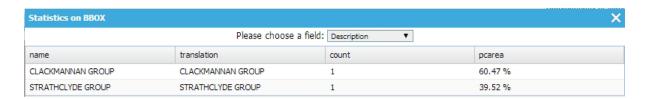

We will cover the Full Statistics option first and then the Lithology Statistics

The Full Statistics option brings up a dialog with collapsible hierarchical lists of geological age and lithology terms from the CGI dictionaries. You can select a number of age values or lithology values or combinations of these. Features satisfying any of the lithology terms and any of the age terms selected will be retrieved; if no age or lithology terms are selected then the features are not filtered on that property. By default, if you select a term that has sub

categories (represented by a folder icon) then all the sub categories will also be selected. This will usually be what you want but you can unselect these afterwards if you wish.

The selected properties are queried for combined with a BBOX filter on the current view port of the portal.

The Lithology Statistics option allows you to select just one lithology term but also to specify a proportion value so that only features where that lithology is present in at least the given proportion are returned. (It checks that the lower value of the proportion range is at least as high as the specified query proportion).

INSPIRE Conformance

In the previous section we have set up, in INSPIRE terminology, both a 'Web Feature Service and Filter Encoding Implementation of Pre-defined Dataset Download Service' and a 'Web Feature Service and Filter Encoding implementation of Direct Access Download Service'. The former needs to conform to the requirements set out in chapter 6 and the latter to the requirements set out in chapter 7 of the Technical Guidance for the implementation of INSPIRE Download Services. We briefly illustrate how the example service conforms to each requirement below. We emphasise that for OneGeology purposes only you do not need to conform to INSPIRE Technical Guidance (TG) Requirements however as most of them are just standard capabilities of competent and useful WFS 2.0 services we include them in this cookbook and point out that INSPIRE requirements 46, 47, 48, 49, 52, 54, 55, 62, 63, 64, 65, 66, 67 and 68 below are what one might expect from a good WFS 2.0 for general use and the GeoServer software described here can provide these capabilities.

Pre-defined Dataset Download

TG Requirement 46

Implementations shall conform to ISO 19142 Conformance Class 'Simple WFS'

The GetCapabilities response from the service includes the declaration

```
<ows:Constraint name="ImplementsBasicWFS">
 <ows:NoValues/>
 <ows:DefaultValue>TRUE</ows:DefaultValue>
</ows:Constraint>
```

and the WFS 2.0.0 standard states that conforming to the Basic WFS class means also implementing the Simple WFS conformance class. Running the OGC CITE tests that check for conformance to the standards is part of the GeoServer build process. This doesn't guarantee full conformance, especially as not all the same tests may be run against app-schema (?). Apart from the queries getting information about what the service holds like GetCapabilities, DescribeFeatureType, ListStoredQueries and DescribeStoredQueries, this conformance class requires that the server at least implements a GetFeature request with the WFS 2.0.0 standard query to get a feature using its id as shown by the example request below:

http://localhost:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetFeature& storedquery_id=urn:ogc:def:query:OGC-WFS::GetFeatureById&id=mf.1&

Note

At the time of writing (Dec 2013) there seems to be a bug so that the above query doesn't actually work. See bug report GEOS-6233

TG Requirement 47

Implementations shall conform to ISO 19143 Conformance Class 'Query'

We can send a query using the wfs:Query element such as the below

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
  "http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd
 http://www.opengis.net/gml/3.2
  http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 service="WFS"
 version="2.0.0"
 outputFormat="application/gml+xml; version=3.2"
 count="10">
<wfs:Query typeNames="gsml:MappedFeature">
</wfs:Query>
</wfs:GetFeature>
```

TG Requirement 48

Implementations shall conform to ISO 19142 Conformance Class 'HTTP Get'

This is illustrated by an example KVP encoded request like the following:

http://localhost:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetFeature&typenames=gsml:MappedFeature&count=20&

TG Requirement 49

Pre-defined Stored Queries shall be provided to make pre-defined datasets available.

The stored query with the name specified in the technical guidance can be invoked with the following query. (Note you must have used the CreateStoredQuery request in the previous chapter to create the stored query first.)

http://localhost:8080/geoserver/ows?service=wfs&version=2.0.0&request=GetFeature&storedquery_id=http://inspire.ec.europa.eu/operation/download/GetSpatialDataSet&DataSetIdCode=13603180&DataSetIdNamespace=http://data.bgs.ac.uk/id/dataHolding/&CRS=urn:ogc:def:crs:EPSG::4326&Language=eng&count=20&

TG Requirement 50

Any possible (i.e. available) combinations of CRS/DataSetIdCode/DataSetIdNamespace/language shall be made available through pre-defined stored queries.

In the example service there is only one combination of CRS/DataSetIDCode/DataSetIdNamespace/language available and that is illustrated under requirement 49 above.

TG Recommendation 13

The following identifier should be used to identify the Stored Query for serving pre-defined Spatial Data Sets: http://inspire.ec.europa.eu/operation/download/GetSpatialDataSet

This has been implemented as illustrated by the query under requirement 49 above.

TG Requirement 51

Pre-defined Stored Queries shall use the parameter names "CRS", "DataSetIdCode", "DataSetIdNamespace" and "Language" to identify the CRS, dataset ID code, dataset ID namespace and language components of a query.

The query shown under requirement 49 above shows the use of these parameters.

TG Requirement 52

A separate WFS endpoint shall be provided for each INSPIRE dataset thus providing one dataset per GetCapabilities response.

Currently there is only one dataset in the example service so this is satisfied trivially.

TG Requirement 53

INSPIRE Metadata for the Download Service shall EITHER be linked to via an <inspire_common:MetadataURL> in an extended capabilities section, OR the extended capabilities section shall contain all the INSPIRE Metadata for the Download Service in accordance with Table 4 and the inspire_dls:ExtendedCapabilities schema.

The example service uses the inspire_common:MetadataURL element as shown below. This has a dummy URL as the example isn't a real service and so it isn't registered in a metadata catalogue. The INSPIRE plugin GUI described in the previous chapter only allows setting a MetadataURL. If you wanted to include all the INSPIRE metadata inside the extended capabilities section itself then you would need to manually edit the xxx file.

```
<inspire_common:MetadataUrl xsi:type="inspire_common:resourceLocatorType">
 <inspire_common:URL>
  http://metadata.bgs.ac.uk/geonetwork/srv/en/csw?SERVICE=CSW&
   REQUEST=GetRecordById&ID=_id_of_service_to_be_entered_here_&
   elementSetName=full&OutputSchema=http://www.isotc211.org/2005/gmd&
 </inspire_common:URL>
 <inspire_common:MediaType>
  application/vnd.ogc.csw.GetRecordByIdResponse_xml
 </inspire_common:MediaType>
</inspire_common:MetadataUrl>
```

TG Requirement 54

A network service [Download Service] metadata response shall contain a list of the natural languages supported by the service. This list shall contain one or more languages that are supported.

This is included in the extended capabilities section of the GetCapabilities response as shown below

```
<inspire_common:SupportedLanguages
 xsi:type="inspire_common:supportedLanguagesType">
 <inspire_common:DefaultLanguage>
  <inspire_common:Language>eng</inspire_common:Language>
 </inspire_common:DefaultLanguage>
</inspire_common:SupportedLanguages>
<inspire_common:ResponseLanguage>
 <inspire_common:Language>eng</inspire_common:Language>
</inspire_common:ResponseLanguage>
```

TG Requirement 55

A client may specify a specific language in a request. If the requested language is contained in the list of supported languages, the natural language fields of the service response shall be in the requested language. It the requested language is not supported by the service, then this parameter shall be ignored.

This is trivially supported by the example service as there is only one language supported and all responses are in that language.

TG Requirement 56

The GetCapabilities operation is extended by an additional parameter that indicates the client's preferred language.

The name of this parameter shall be "LANGUAGE". The parameter values are based on ISO 639-2/B alpha 3 codes as used in [INS MDTG].

For the example service the parameter value is 'eng' which is the ISO 639-2/B alpha 3 code for English.

TG Requirement 57

If a client request specifies an unsupported language, or the parameter is absent in the request, the above fields [Title, Abstract] shall be provided in the service default language.

The response to the following request specifying the unsupported Finnish language is in English:

http://localhost:8080/geoserver/ows?service=wfs&version=2.0.0&
request=GetCapabilities&language=fin&

TG Requirement 58

The Extended Capabilities shall indicate the response language used for the GetCapabilities-Response: Depending on the requested language the value of the <inspire_common:ResponseLanguage> corresponds to the current used language. If a supported language was requested, <inspire_common:ResponseLanguage> shall correspond to that requested language. If an unsupported language was requested or if no specific language was requested <inspire_common:ResponseLanguage> shall correspond to the service default language <inspire_common:DefaultLanguage>

This is illustrated by the fragment shown under requirement 54.

TG Requirement 59

The Extended Capabilities shall contain the list of supported languages indicated in <inspire_common:SupportedLanguages>. This list of supported languages shall consist of 1. exactly one element <inspire_common:DefaultLanguage> indicating the service default language, and 2. zero or more elements <inspire_common:SupportedLanguage> to indicate all additional supported languages. Regardless of the response language, the list of supported languages is invariant for each GetCapabilities-Response.

This is illustrated by the fragment shown under requirement 54.

TG Requirement 60

The Extended Capabilities shall use the XML Schema as defined in the INSPIRE online schema repository.

The example service declares the above schema in the schemaLocation attribute of the GetCapabilities response as shown below and the response document validates according to the Schema.

```
xsi:schemaLocation=
"http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2.0/wfs.xsd
 http://inspire.ec.europa.eu/schemas/common/1.0
 http://inspire.ec.europa.eu/schemas/common/1.0/common.xsd
 http://inspire.ec.europa.eu/schemas/inspire_dls/1.0
 http://inspire.ec.europa.eu/schemas/inspire_dls/1.0/inspire_dls.xsd"
```

TG Recommendation 14

For further language support for other operations it is recommended to replace the operation-online-resources in each language specific GetCapabilities-Response by a specific operation-online-resource for that language. To support the additional operation-online-resources the service shall listen at the language specific operation end-points to distinguish for the requested languages.

The example service does not implement this recommendation as it doesn't support more than one language anyway.

TG Recommendation 15

The support of IETF RFC 4646 is recommended wherever the support of ISO/639 B alpha3 for languages infringes the conformity with the standard used for implementing the [INS NS].

I think this doesn't apply to any INSPIRE service as yet as there is no language support defined in current OGC standards applying to WFS.

Direct Access Download

TG Requirement 61

Implementations shall meet TG Requirement 48 (conformance to [ISO 19142] 'HTTP GET' conformance class) and TG Requirement 52 (one endpoint for each INSPIRE dataset).

This is covered under requirements 48 and 52 above.

TG Requirement 62

Implementations shall conform to ISO 19142 Conformance Class 'Basic WFS'.

The comments under requirement 46 in fact also apply to this requirement. A illustration of a query showing GetFeature with a Query element which is part of the Basic WFS conformance class but not the Simple WFS conformance class would be the BBOX query below

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
  "http://www.opengis.net/wfs/2.0
   http://schemas.opengis.net/wfs/2.0/wfs.xsd
  http://www.opengis.net/gml/3.2
   http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 service="WFS"
 version="2.0.0"
 outputFormat="application/gml+xml; version=3.2"
 count="10">
<wfs:Query typeNames="gsml:MappedFeature">
 <fes:Filter>
  <fes:BBOX>
   <fes:ValueReference>gsml:shape</fes:ValueReference>
   <gml:Envelope>
    <gml:lowerCorner>-1.0 51.0</gml:lowerCorner>
    <gml:upperCorner>0.0 52.0</gml:upperCorner>
   </gml:Envelope>
  </fes:BBOX>
 </fes:Filter>
```

```
    </wfs:Query>
</wfs:GetFeature>
```

TG Requirement 63

A Direct Access Download Service shall conform to ISO 19143 'Ad hoc Query' Conformance Class.

The age, lithology and BBOX queries described in the previous chapter show various examples of different queries in this class.

TG Requirement 64

A Direct Access Download Service shall conform to ISO 19143 'Resource Identification' Conformance Class.

This is illustrated by the query below

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=
  "http://www.opengis.net/wfs/2.0
    http://schemas.opengis.net/wfs/2.0/wfs.xsd
  http://www.opengis.net/gml/3.2
    http://schemas.opengis.net/gml/3.2.1/gml.xsd"
 xmlns:gsml="http://xmlns.geosciml.org/GeoSciML-Core/3.2"
 xmlns:fes="http://www.opengis.net/fes/2.0"
 xmlns:wfs="http://www.opengis.net/wfs/2.0"
 xmlns:gml="http://www.opengis.net/gml/3.2"
 service="WFS"
 version="2.0.0"
 outputFormat="application/gml+xml; version=3.2"
 count="10">
<wfs:Query typeNames="gsml:MappedFeature">
 <fes:Filter>
  <fes:ResourceId rid="mf.1"/>
 </fes:Filter>
</wfs:Query>
</wfs:GetFeature>
```

TG Requirement 65

A Direct Access Download Service shall conform to ISO 19143 'Minimum Standard Filter' Conformance Class.

This is illustrated by the age and lithology queries in the previous chapter.

TG Requirement 66

A Direct Access Download Service shall conform to ISO 19143 'Minimum Spatial Filter' Conformance Class.

This is illustrated by the BBOX query in the previous chapter.

TG Requirement 67

A Direct Access Download Service shall conform to ISO 19143 'Minimum Temporal Filter' Conformance Class.

The GetCapabilities response states that this is supported as is the 'During' filter operation but the example data does not contain any temporal properties.

TG Requirement 68

A Direct Access Download Service shall conform to ISO 19143 'Minimum XPath' Conformance Class.

The age and lithology queries in the previous chapter show the use of XPath to refer to some nested property values.