

# Developer Documentation



*How to generate and run GeoNetwork  
for*

*Bundesamt für Strahlenschutz*



Author: GeoCat bv  
Last Updated: 10/8/2015

# Table of Contents

## [1. Set up local environment](#)

### [1.1. Prerequisites](#)

### [1.2. Clone Git Repository](#)

#### [1.2.1. Clone Git Submodules](#)

## [2. Generate war](#)

## [3. Configure and Deploy](#)

### [3.1. Configure using the Source Code](#)

### [3.2. Configure using a provided war](#)

### [3.3. Deployment](#)

## [4. \[Optional\] Upgrade with Upstream changes](#)

## [5. Metadata management](#)

### [5.1. Create metadata](#)

### [5.2. Import existing Metadata](#)

### [5.3. Upgrade metadata](#)

### [5.4. Export metadata to JSON](#)

### [5.5. Configure INSPIRE for iso19139](#)

## [6. Modify Bfs Schema](#)

### [6.1. Update labels/recommended values](#)

# Document updates

| Version | Date      | Author               | Description   |
|---------|-----------|----------------------|---|
| 1.0     | 10/8/2015 | María Arias de Reyna | Initial version   |
| 1.1     | 9/9/2015  | Jose García          | Section to update labels and recommended values   |
| 1.2     | 2/10/2015 | Jose García          | Sections to upgrade and create the metadata and reorganize the document sections related to metadata management |
| 1.3     | 2/1/2015  | Jose García          | INSPIRE section   |

# Table of Contents

## [1. Set up local environment](#)

### [1.1. Prerequisites](#)

### [1.2. Clone Git Repository](#)

#### [1.2.1. Clone Git Submodules](#)

## [2. Generate war](#)

## [3. Configure and Deploy](#)

### [3.1. Configure using the Source Code](#)

### [3.2. Configure using a provided war](#)

### [3.3. Deployment](#)

## [4. \[Optional\] Upgrade with Upstream changes](#)

## [5. Metadata management](#)

### [5.1. Create metadata](#)

### [5.2. Import existing Metadata](#)

### [5.3. Upgrade metadata](#)

### [5.4. Export metadata to JSON](#)

### [5.5. Configure INSPIRE for iso19139](#)

## [6. Modify Bfs Schema](#)

### [6.1. Update labels/recommended values](#)

# 1. Set up local environment

## 1.1. Prerequisites

To set up the local environment you have to have installed on your system the software **Git**, **Maven** version 3+<sup>1</sup> and a **Java JDK** version 7+.

## 1.2. Clone Git Repository

The first thing to do is locate the repository url where your customized version of GeoNetwork is stored. For example, if you are going to use the version stored on GeoCat's repository, the url will be ***git@eos.geocat.net:bfs/layer-metadata-profile.git***

If we are using the command line, we just have to place the terminal on the folder where we are going to explode the GeoNetwork source code (\$path) and run:

```
cd $path  
git clone git@eos.geocat.net:bfs/layer-metadata-profile.git
```

This command will download from the repository not only the plain source code but also the git configuration and part of the history of the repository. The output of the terminal should look like this:

```
$ git clone git@eos.geocat.net:bfs/layer-metadata-profile.git  
Cloning into 'layer-metadata-profile'...  
remote: Counting objects: 166664, done.  
remote: Compressing objects: 100% (57140/57140), done.  
remote: Total 166664 (delta 91319), reused 165825 (delta 90728)  
Receiving objects: 100% (166664/166664), 155.49 MiB | 4.07 MiB/s, done.  
Resolving deltas: 100% (91319/91319), done.  
Checking out files: 100% (9493/9493), done.
```

### 1.2.1. Clone Git Submodules

GeoNetwork contains several submodules that are usually not cloned by default. To make sure we have the latest version available of this submodules on our local source code, we have to run the “submodule init” and a “submodule update” git commands.

If we are using the command line, we just have to place the terminal on the root folder of the GeoNetwork source code and run:

```
cd $path/layer-metadata-profile;  
git submodule init;  
git submodule update
```

---

<sup>1</sup> Official Documentation available at <https://maven.apache.org/install.html>

After a similar output than on the previous step, we will have all the source code available on our local machine, ready to be built:

```
$ git submodule init; git submodule update
Submodule 'e2e-tests/chromedriver' (https://github.com/geonetwork/chromedriver.git)
registered for path 'e2e-tests/chromedriver'
Submodule 'geoserver' (https://github.com/geonetwork/core-geoserver.git) registered for
path 'geoserver'
Submodule 'web-ui/src/main/resources/catalog/lib/style/bootstrap'
(https://github.com/twbs/bootstrap.git) registered for path
'web-ui/src/main/resources/catalog/lib/style/bootstrap'
Submodule 'web-ui/src/main/resources/catalog/lib/style/font-awesome'
(https://github.com/FortAwesome/Font-Awesome.git) registered for path
'web-ui/src/main/resources/catalog/lib/style/font-awesome'
Cloning into 'e2e-tests/chromedriver'...
remote: Counting objects: 21, done.
remote: Total 21 (delta 0), reused 0 (delta 0), pack-reused 21
Unpacking objects: 100% (21/21), done.
Submodule path 'e2e-tests/chromedriver': checked out
'501f7852e8620c4f33de37cf0911f73bf2d2d002'
Cloning into 'geoserver'...
remote: Counting objects: 460, done.
remote: Total 460 (delta 0), reused 0 (delta 0), pack-reused 460
Receiving objects: 100% (460/460), 62.82 MiB | 9.90 MiB/s, done.
Resolving deltas: 100% (169/169), done.
Submodule path 'geoserver': checked out
'1eb4c68418e44ac36da5cb425124118799f75e49'
Cloning into 'web-ui/src/main/resources/catalog/lib/style/bootstrap'...
remote: Counting objects: 73124, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 73124 (delta 1), reused 0 (delta 0), pack-reused 73120
Receiving objects: 100% (73124/73124), 85.84 MiB | 10.51 MiB/s, done.
Resolving deltas: 100% (44536/44536), done.
Submodule path 'web-ui/src/main/resources/catalog/lib/style/bootstrap': checked out
'a365d8689c3f3cee7f1acf86b61270ecca8e106d'
Cloning into 'web-ui/src/main/resources/catalog/lib/style/font-awesome'...
remote: Counting objects: 32710, done.
remote: Compressing objects: 100% (410/410), done.
remote: Total 32710 (delta 325), reused 0 (delta 0), pack-reused 32300
Receiving objects: 100% (32710/32710), 18.32 MiB | 6.01 MiB/s, done.
Resolving deltas: 100% (17591/17591), done.
Submodule path 'web-ui/src/main/resources/catalog/lib/style/font-awesome': checked out
'2649d91d18629bab071449b3bc4cb02761037a57'
```

## 2. Generate war

Once you have the source code on your local machine, you can invoke maven to run. If we are using the command line, we just have to place the terminal on the root folder of the GeoNetwork source code and run:

```
export MAVEN_OPTS="-Xmx512M -XX:MaxPermSize=256M"  
mvn clean package install -DskipTests
```

Be patient, because this command may take long to finish, as it has to download from maven artifacts repository many third party libraries. When the build finishes, you should see something like this:

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 04:43 min  
[INFO] Finished at: 2015-07-28T13:16:36+01:00  
[INFO] Final Memory: 194M/407M  
[INFO] -----
```

The generated war file we are interested in is located on **web/target/geonetwork.war**.

[Optional] You can try the build by moving to the web folder and run a Jetty server from there:

```
cd web; mvn jetty:run; cd ..
```

You can test your deployment by opening a browser and entering <http://localhost:8080/geonetwork>

## 3. Configure and Deploy

There are many things that can be configured in GeoNetwork<sup>2</sup>, but the basic thing to configure on a deployment is the database GeoNetwork is going to use.

### 3.1. Configure using the Source Code

First check which type of database is used. Choose the database type to use on the file **web/src/main/webapps/WEB-INF/config-node/srv.xml**.

Then update the **web/src/main/webapps/WEB-INF/config-db/jdbc.properties** file with connection information.

If we have done changes on the configuration files, we have to rebuild the war file. This is done by invoking again the **mvn package install -DskipTests** command. Make sure you run it on the root folder of GeoNetwork's source code.

---

<sup>2</sup> See <http://geonetwork-opensource.org/manuals/trunk/eng/users/administrator-guide/index.html> for more information.

## 3.2. Configure using a provided war

To configure the database, you have to modify a couple of files contained inside the war file. Remember that a war file is nothing more than a compressed zip file with the extension “.war”.

First check which type of database is used. Choose the database type to use on the file **WEB-INF/config-node/srv.xml**. Then update the **WEB-INF/config-db/jdbc.properties** file with connection information.

Once you have modified this files, make sure you place them again inside the **geonetwork.war** file.

## 3.3. Deployment

To deploy your GeoNetwork instance, copy the **geonetwork.war** file to your application server. In the case of Tomcat, there should be a folder called **webapps** which contains all the applications to deploy. This is the folder where the **geonetwork.war** file should be copied in.

Once the application server is running, you have to make sure the metadata profiles you are going to use are enabled on your installation. To do this, you have to be logged in as an administrator user. The default administrator user is *admin/admin*.

After login, you have to enter the administration interface by clicking on the “**Admin Console**” on the toolbar. Then you select the “**Metadata and Templates**” option, which will display the list of schemas already available to be enabled. On that list, you can select the schemas you want to enable on your installation. You can select more than one schema at the same time.



Metadata & templates

Formatter

Schematron

Load samples and templates for metadata standards

## Standards available

1 selected ▼

### Dublin Core - CSW

Metadata records produced by CSW services.

### Dublin Core

The Dublin Core metadata standard

### Geographic information - Methodology for feature cataloguing (ISO 19110:2005)

ISO 19110 standard for describing Feature Types

### Geographic information - Metadata (ISO/TS 19139:2007)

ISO19139 metadata standard

### BFS - layer metadata

BFS - layer metadata standard

Load templates for selected standards

Load samples for selected standards


Once you have selected all the schemas you plan to use, you can click on both buttons at the bottom of the list. The first one will load the templates, which will be used when you create a new metadata based on those schemas. The second button (optional) will load the metadata example records for the schemas, which will be added to the catalog. You can remove this examples later if you want.

Now that you have your schema profiles enabled on the server, you can create your own metadata based on this schemas.

To create a new metadata, click on the “**Contribute**” link of the top toolbar. This will lead you to the editor dashboard, where you will be able to see all the current metadata records and templates. If you click on the “**Add new record**” button, you will be redirected to a new view where you can select the type of metadata you want to create. First, select the type of metadata and then select the template that fits better with your use case.

## Create a

Create a  
**Others**

  
Others

From **Vector - Layer Template**

Generic Layer Template

**Vector - Layer Template**

WMS - Layer Template

WMTS - Layer Template

+ Create

▼

When you click on the “**Create**” button, you will be redirected to the metadata editor which will display your newly created metadata based on the template you have chosen.

## 4. [Optional] Upgrade with Upstream changes

You will probably want to keep your source code upgraded with the latest upstream changes. The easiest way to achieve this is to add a new remote to git with the official GeoNetwork repository and do frequent pulls from it.

So, we add a new remote to our local git repository:

```
git remote add upstream https://github.com/geonetwork/core-geonetwork.git
```

Now, check the version of GeoNetwork on which your source code is based in. This is defined on the pom.xml files of your source code. At the moment of writing this documentation, the source code is based on version 3.0.1, which means that we will want to keep upgraded with the **3.0.x** branch of GeoNetwork repository.

So, to pull the changes from upstream and merge them to our source code, we run the following command:

```
git pull upstream 3.0.x
```

if there are no conflicts, we will be directly redirected to the commit message we want to use to mark this merge. If there are conflicts, we will have to solve them first before being able to mark the pull as merged.

Once the conflicts (if any) are solved and the commit is done, we can push to our repository the changes, so the merge is not only stored on our local machine:

```
git push
```

If you are using GeoCat's repository as the main repository (see *Clone Git Repository* at the beginning of this document), check that you have privileges to write to the repository. If you don't, the push will fail.

Don't forget to test the software before re-deploying it. Even when git doesn't detect conflicts, that doesn't mean that the code is not broken.


## 5. Metadata management

### 5.1. Create metadata

To create a new metadata, click on the “**Contribute**” link of the top toolbar. This will lead you to the editor dashboard, where you will be able to see all the current metadata records and templates. If you click on the “**Add new record**” button the following form is displayed to select the metadata template to use:

Create a

Create a  
**Others**

  
Others

From **Vector - Layer Template**

Generic Layer Template

**Vector - Layer Template**

WMS - Layer Template

WMTS - Layer Template

+ Create

▼

Select the metadata template and click the button **Create** to open the metadata editor and enter the metadata information:

🗺️ 🏠 ⚙️ ✓ Validate ↺ Cancel 📄 Save & close 💾 Save metadata 👁️

**▼ Layer Information**

Title \*

INSPIRE Identifier \*

**▼ Layer Type**

**▼ WMS**

Host \*

Path \*

Layer \*

Transparent \* ☒

Version \*

Styles \*

Format \*  Recommended values ▾

**▼ Download**

🔗 Associated resources +

✓ Validation ↺

[Need help](#)

- Layer Information
- Layer Type
- Download
- WFS
- Layer Filters
- Value Filter
- Point in Time Filter
- OpenLayers Properties
- Property
- Timeseries Chart
- Properties
- Bar Chart Properties
- Metadata

To validate the metadata click the button **Validate**:

🔔 Validation
👍 👎 ↺

**XSD Validation Error**

0 Error

**Schematron validation / Bfs rules**

0 / 1

--

Duplicated Time Series Chart Property with name 'titleTpl'.

**Recommendations**

0 / 0

**ISO rules**

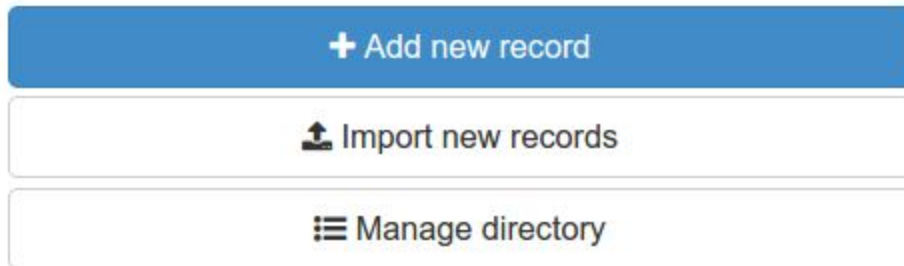
0 / 0

The validation report contains a section Schematron validation /BfS rules that verify that the property names defined in the different sections: OpenLayers, Time Series Chart and Bar Chart are not duplicated.

To save the metadata and finish the editing session click **Save & Close**.

## 5.2. Import existing Metadata

An editor can import metadata in the catalog file in different formats: XML, MEF or ZIP. After login, go to the contribute page (<http://.../srv/eng/catalog.edit>) and select the “**Import new records**” button:



The import new records page allows you to import records in three ways:

- choose **Upload a file from your computer** and choose one XML or MEF file to load
- choose **Copy/Paste** and copy the XML document in the textarea
- choose **Import a set of files from a folder on the server** and set the path of the folder in the server

To import multiple file at a time, use the MEF format or the import on the server options.

After defining the type of import, configure the other import settings:

Import new records

☒ Upload a file from your computer  
☐ Copy/Paste  
☐ Import a set of files from a folder on the server

+ Choose or drop resource here

**Type of file**    ☒ XML    ☐ MEF

**Type of record**    Metadata ▼

**Record identifier processing**

☒ None  
☐ Overwrite metadata with same UUID  
☐ Generate UUID for inserted metadata

**Apply XSLT conversion**    ▼

☐ Validate

☐ Assign to current catalog

**Assign to group**    ▼

**Assign to category**    ▼

+ Import

- Type of file: when uploading or loading file from the server, define the type of file to load. It could be XML for importing XML documents or MEF (equivalent to ZIP) for importing MEF format.
- Type of record:
  - *Metadata* should be used when loading a normal metadata record
  - *Template* should be used when the loaded metadata record will be used as a template.
- Record identifier processing determines how to handle potential clashes between the UUID of the record loaded and UUIDs of metadata records already present in the catalog. 3 strategies are available:

- *None*: the UUID of the record loaded is left unchanged. If a metadata record with the same UUID is already present in the catalog, an error message is returned.
- *Overwrite metadata with same UUID*: any existing metadata record in the catalog having the same UUID as the loaded record will be updated.
- *Generate UUID for inserted metadata*: a new UUID is affected to the loaded record. No metadata will be overwritten.
- Apply XSLT conversion allows to transform the record loaded using an XSLT stylesheet. A list of predefined transformations is provided. The selected transformation should be compatible with the standard of the loaded record.
- Validate trigger the validation of the record before loading it. In case of error the record is rejected and an error reported. If not selected, the metadata can be validated afterwards through the editor.
- Assign to current catalog assign the current catalog as origin for the record, in case the MEF file indicate another source.
- Assign to Group define the group of the loaded record. This is important because it will define the default privileges over the metadata.
- Assign to Category define a local category to assign to the loaded record.

Click **“import”** to start the import process. After processing, a summary is provided with the following details:

- the total count of imported metadata
- errors messages
- if only one record is imported, a link to that record is provided.

### 5.3. Upgrade metadata

During the development of the project have been done different changes in the metadata schema. To allow upgrading the metadata create with initial versions of the schema to the latest version has been develop a xslt process.

The process is defined in the following file:

***schemas/iso19139.bfs/src/main/plugin/iso19139.bfs/process/upgrade-metadata.xsl***

Upgrading these elements:

- bfs:MD\_VectorLayerType
- bfs:MD\_PointInTimeFilter
- bfs:MD\_TimeRangeFilter
- bfs:MD\_ValueFilter

- bfs:TimeInstant
- bfs:download

To make the process available in the UI, is required to update this file:

***web-ui/src/main/resources/catalog/config/batch-process-cfg.json***

to add a new config section (the change is already done, but requires to be preserved when upgrading GeoNetwork):

```
{
  "key": "upgrade-metadata",
  "type": "fixed-params",
  "params": [
  ]
}
```

To execute the process from the UI, login as Administrator in GeoNetwork and go to **Administration > Tools > Batch process**.

In the form filter and select the metadata to upgrade. Once selected, in the **Configure a process** section, select the process **upgrade-metadata** and click **Run** to upgrade it.



[Index admin](#)
[Batch process](#)
[Transfert ownership](#)

Select records to process

☒ Metadata
 ☒ Template
 ☐ Directory entry

Search ...
 

Q

☒ odl\_brutto\_24h  
  
☐ Vector - Layer Template - Test 1  
  
☐ WMS - Layer Template  
  
☐ Vector - Layer Template  
  
☐ Vector - Layer Template  
  
☐ WMS - Layer Template  
  
☐ WMTS - Layer Template  
  
☐ Generic Layer Template

Group

Owner

Category

1 - 8 on 8 < >

Configure a process

upgrade-metadata

▶ Run

## 5.4. Export metadata to JSON

Specifically for this project we have added a new service that exports the metadata from XML to JSON. This service can be found on the following URL:

***[http://\\$yourServerUrl/geonetwork/srv/eng/xml\\_iso19139Tojson?uuid=\\$UUID](http://$yourServerUrl/geonetwork/srv/eng/xml_iso19139Tojson?uuid=$UUID)***

This service accepts as parameter the unique file identifier of the metadata (\$UUID). This identifier is shown on the metadata view as Identifier and it is also placed inside the metadata record on the following xpath:

***`/bfs:MD_Metadata /gmd:fileIdentifier/gco:CharacterString`***

Example of output:

```

{
  "dspTxt": "Brutto-ODL-Messfahrt",
  "inspireId": "mobil_brutto",
  "defaultStart": "2015-06-22 08:20:22",
  "dateFormat": "%Y-%m-%d %H:%i:%s",
}

```

16

Veenderweg 13, 6721 WD Bennekom - The Netherlands +31(0)318416664  
[info@GeoCat.net](mailto:info@GeoCat.net) - [www.GeoCat.net](http://www.GeoCat.net) CoC#: 08162601 - VAT#: NL818319720.B01

```

"mgr": "{id:909,name:'ODL-Brutto'}",
"defaultMgr": "909",
"uwblid": null,
"filter": { "type": "daterange", "interval": 1440, "unit": "minute", "vendorParamStart": "TIME", "vendorParamEnd": "mgr" },
"layerConfig": {
  "wms": { "url": "http://test-gis3-fr.lab.bfs.de:8080/geoserver/imis/wms?",
    "layers": "odl_brutto_mobil",
    "transparent": true,
    "version": "1.1.1",
    "styles": null, "format": "image/png",
    "timeExtendStart": "2015-05-15 00:00:00",
    "timeExtendEnd": "2015-06-30 00:00:00" },
  "download": null,
  "openLayers": { "isBaseLayer": false,
    "singleTile": true,
    "ratio": 1.0,
    "buffer": 1.0,
    "displayInLayerSwitcher": true,
    "attribution": null,
    "units": null,
    "maxResolution": null,
    "minResolution": null,
    "numZoomLevels": null,
    "minScale": null,
    "maxScale": null },
  "graphic": null }
},
{ "dspTxt": "Brutto-ODL-10-min",
  "inspireId": "odl_brutto_10min",
  "defaultStart": "2015-05-05 11:20:00",
  "dateFormat": "%Y-%m-%d %H:%i:%s",
  "mgr": "{id:1,name:'BfS & KFÜ'}, {id:2,name:'BfS'}, {id:3,name:'KFÜ'}",
  "defaultMgr": "1",
  "uwblid": null,
  "filter": { "type": "date",
    "interval": 10,
    "unit": "minute",
    "vendorParamStart": "TIME",
    "vendorParamEnd": "Netz" },
  "layerConfig": { "wms": { "url": "http://test-appsrv1-fr.lab.bfs.de/cgi-imis/wms_imis?",
    "layers": "odl_brutto_10min",
    "transparent": true,
    "version": "1.1.1",
    "styles": null,
    "format": "image/png",
    "timeExtendStart": "2014-08-14 06:10:00",
    "timeExtendEnd": "2015-05-05 11:20:00" },
    "download": null,
    "openLayers": { "isBaseLayer": false,
      "singleTile": true,
      "ratio": 1.0,
      "buffer": 1.0,
      "displayInLayerSwitcher": true,
      "attribution": null,
      "units": null,
      "maxResolution": null,

```

```

        "minResolution":null,
        "numZoomLevels":null,
        "minScale":null,
        "maxScale":null},
    "graphic":null}
},
{"dspTxt":"UV-Messstationen",
"inspireId":"8b100b7e-6a66-40eb-8a7f-5b451733e3d3",
"defaultStart":null,
"dateFormat":null,
"filter":null,
"layerConfig":{"wms":{"url":"http://test-appsrv1-fr.lab.bfs.de/cgi-public/wms_inspire_uv?",
"layers":"ef.environmentalmonitoringfacility",
"transparent":true,
"version":"1.1.1",
"styles":"EF.EnvironmentalMonitoringFacilities.Default",
"format":"image/png",
"timeExtendStart":null,
"timeExtendEnd":null},
"download":{"url":"http://test-appsrv1-fr.lab.bfs.de/deegree-webservices/services/wfs_uv_messstationen?service=WFS&request=GetFeature&VERSION=2.0.0&TYPENAME=ef:EnvironmentalMonitoringFacility&outputformat=application%2Fgml%2Bxml%3B version%3D3.2&resolveDepth=*",
"placeholderStart":null,
"placeholderEnd":null},
"openLayers":{"isBaseLayer":false,
"singleTile":true,
"ratio":1.0,
"buffer":1.0,
"displayInLayerSwitcher":true,
"attribution":null,
"units":null,
"maxResolution":null,
"minResolution":null,
"numZoomLevels":null,
"minScale":null,
"maxScale":null}
}}
}

```

## 5.5. Configure INSPIRE for iso19139

To configure INSPIRE for iso19139 metadata the following steps should be done.

**To prepare a war package with the changes:**

- 1) **Metadata editor.** In schemas/iso19139/src/main/plugin/iso19139/layout/config-editor.xml

Remove the attribute **disabled** from the following line to enable the INSPIRE view in the metadata editor:

```
<view name="inspire" upAndDownControlHidden="true" disabled="true">
```

- 2) **Schematron rules:**

schemas/iso19139/src/main/plugin/iso19139/schematron/schematron-rules-inspire.disabled.sch,  
schemas/iso19139/src/main/plugin/iso19139/schematron/schematron-rules-inspire-strict.disabled.sch

Rename these files to:

- schemas/iso19139/src/main/plugin/iso19139/schematron/schematron-rules-inspire.sch
- schemas/iso19139/src/main/plugin/iso19139/schematron/schematron-rules-inspire-strict.sch

### 3) **Thesaurus files.** Download the following files to this folder:

- web/src/main/webapp/WEB-INF/data/config/codelist/external/thesauri/theme

GEMET:

- i) <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/gemet.rdf>

GEMET-THEME:

- ii) <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/gemet-theme.rdf>

INSPIRE SERVICE TAXONOMY:

- iii) <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/inspire-service-taxonomy.rdf>

### 4) **CSW INSPIRE end-point.**

Change in web/src/main/webapp/WEB-INF/classes/setup/sql/data/data-db-default.sql

```
INSERT INTO Settings (name, value, datatype, position, internal) VALUES
('system/inspire/enable', 'false', 2, 7210, 'y');
```

to

```
INSERT INTO Settings (name, value, datatype, position, internal) VALUES
('system/inspire/enable', 'true', 2, 7210, 'y');
```

For additional configuration of the INSPIRE GetCapabilities go to web/src/main/webapp/xml/csw/capabilities\_inspire.xml and edit the content required.

See additional information in:

<http://geonetwork-opensource.org/manuals/trunk/eng/users/administrator-guide/configuring-the-catalog/inspire-configuration.html?highlight=inspire>

## To update an installation to enable INSPIRE:

### 1) **Metadata editor.** In the file:

TOMCAT\_DIR/webapps/geonetwork/WEB-INF/data/config/schema\_plugins/iso19139/layout/config-editor.xml

Remove the attribute **disabled** from the following line to enable the INSPIRE view in the metadata editor:

```
<view name="inspire" upAndDownControlHidden="true" disabled="true">
```

### 2) **Schematron validation rules.**

TOMCAT\_DIR/webapps/geonetwork/WEB-INF/data/config/schema\_plugins/iso19139/schematron/schematron-rules-inspire.disabled.sch,

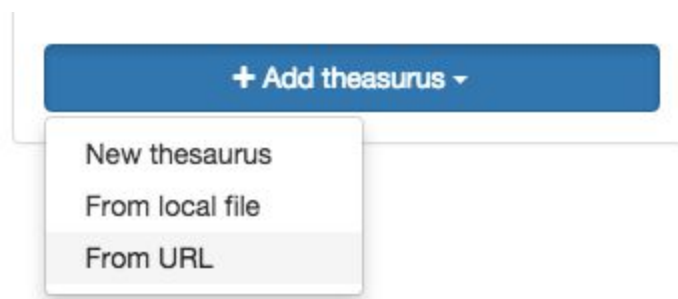
TOMCAT\_DIR/webapps/geonetwork/WEB-INF/data/config/schema\_plugins/iso19139/schematron/schematron-rules-inspire-strict.disabled.sch

Rename these files to:

- TOMCAT\_DIR/webapps/geonetwork/WEB-INF/data/config/schema\_plugins/iso19139/schematron/schematron-rules-inspire.sch
- TOMCAT\_DIR/webapps/geonetwork/WEB-INF/data/config/schema\_plugins/iso19139/schematron/schematron-rules-inspire-strict.sch

**IMPORTANT:** After 1) and 2) GeoNetwork requires to be restarted.

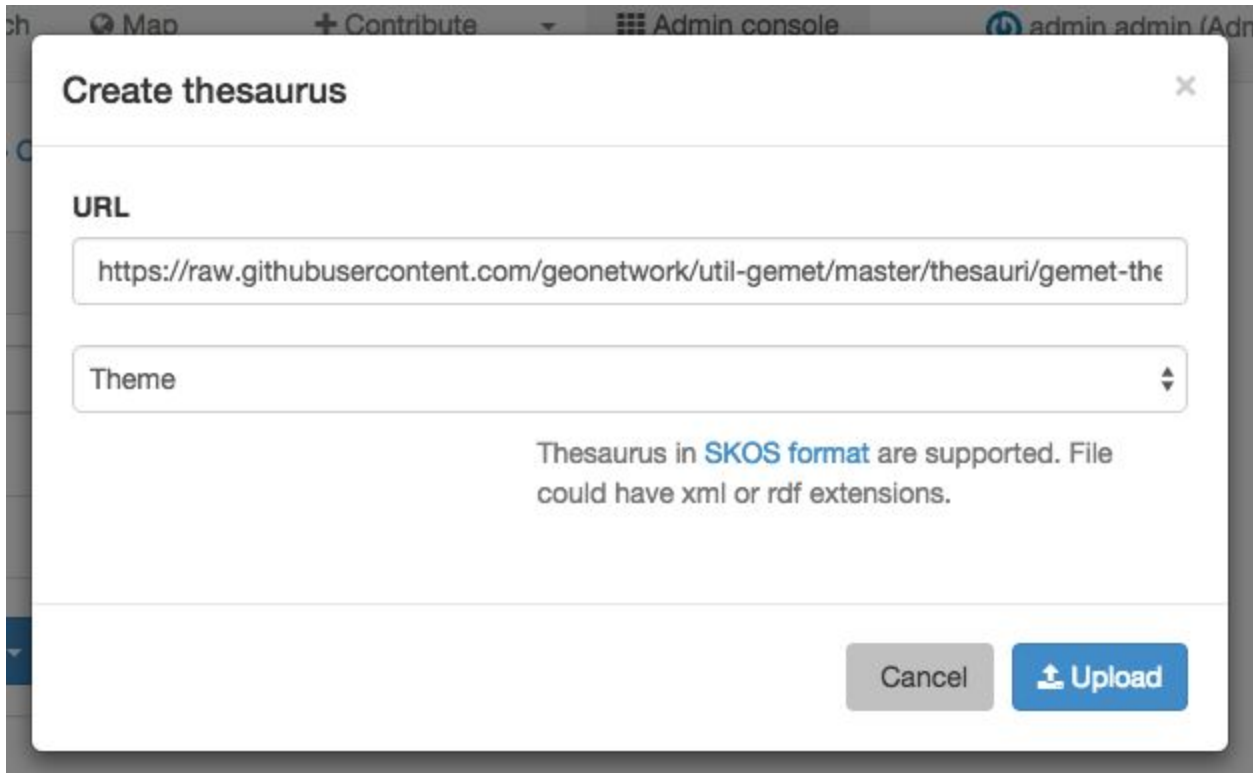
### 3) **Load the INSPIRE thesaurus files.** Go to Admin console > Classification Systems and select the option to Add a thesaurus from URL:



Specify the following thesaurus URLs and the theme

- GEMET:
  - <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/gemet.rdf>

- GEMET-THEME:
  - <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/gemet-theme.rdf>
- INSPIRE SERVICE TAXONOMY:
  - <https://raw.githubusercontent.com/geonetwork/util-gemet/master/thesauri/inspire-service-taxonomy.rdf>



#### 4) CSW INSPIRE end-point.

Go to Admin console > Settings and enable the INSPIRE setting:

##### INSPIRE Directive configuration

INSPIRE ☒

Enable INSPIRE CSW (ie. language support and INSPIRE GetCapabilities document) and INSPIRE indexing. INSPIRE themes thesaurus MUST be installed to properly index themes and annexes.

For additional configuration of the INSPIRE GetCapabilities go to `TOMCAT_DIR/webapps/geonetwork/xml/csw/capabilities_inspire.xml` and edit the content required.

See additional information in:

<http://geonetwork-opensource.org/manuals/trunk/eng/users/administrator-guide/configuring-the-catalog/inspire-configuration.html?highlight=inspire>

## 6. Modify Bfs Schema

On this section we will describe how to add a new field on the bfs schema. As an example, we will show how to add a new element on the schema.

The first step is to modify the schema. The schema is on the folder **schemas/iso19139.bfs** on the source code. Inside this folder, we open the file **src/main/plugin/iso19139.bfs/schema/bfs/bfs.xsd**. There, we search for the parent element where we want to add a new element.

You can use a visual editor of the XSD schema or a text editor. In both cases, make sure that you create a valid XSD schema.

For example, if you want to add a parameter called `bfs:paramAlias` as a child of `bfs:filter`, you should look for the `MD_ValueFilterType` complexType definition and add a new element like this:

```
<xs:sequence>

    <xs:element name="paramName" type="gco:CharacterString_PropertyType"/>

    <xs:element name="paramAlias" type="gco:CharacterString_PropertyType"/>

    <xs:element name="value" type="gco:CharacterString_PropertyType" minOccurs="1"
maxOccurs="unbounded"/>
```

The only thing missing now on the schema would be a proper label translation for this field. Search for the `labels.xml` files placed on `src/main/plugins/iso19139.bfs/loc/${LANG}/labels.xml` and add an element entry for the new type:

```
<element name="bfs:paramAlias">

    <label>Alias</label>

    <description></description>

</element>
```

Once we have modified the schema, we should open the file **src/main/plugin/iso19139.bfs/layout/config-editor.xml**. This file defines the metadata editor fields to display. On this case, we are displaying all children of `bfs:filter`, so we don't have to add the new field to the editor explicitly. But if we had to, we just have to add a new field element inside the view element `"default"` which is the one showing all the bfs fields.

The last step is to modify the service that converts the xml data to json (described on the following section). To update the service, search for the file **src/main/webapp/xsl/conversion/export/xml\_json.xsl** on the folder `web` located at the root folder of the source code. Here, you have to add the new field on the corresponding template. On this case, look again for the match defined as the parent element (`MD_ValueFilterType`) and add a new child:

```
<xsl:template match="bfs:MD_ValueFilter">
  "filter":{
    "type":"value",
    "param": "<xsl:value-of select='bfs:paramName/gco:CharacterString' />",
    "alias": "<xsl:value-of select='bfs:paramAlias/gco:CharacterString' />",
    <xsl:for-each select="bfs:value">
      "value":<xsl:value-of select="gco:CharacterString" />,
    </xsl:for-each>
    "defaultValue": "<xsl:value-of select='bfs:defaultValue/gco:CharacterString' />",
    "allowMultipleSelect": "<xsl:apply-templates select='bfs:allowMultipleSelect/gco:Boolean' />"
  },
</xsl:template>
```

## 6.1. Update labels/recommended values

The labels and recommended values are stored in the file **schemas/iso19139.bfs/src/main/plugin/iso19139.bfs/labels/eng/labels.xml**.

**Example of a standard label:**

```
<element name="bfs:tilematrixset">
  <label>Tile Matrix Set</label>
  <description></description>
</element>
```

**Example of a label with recommended values:**

```
<element name="bfs:format">
  <label>Format</label>
  <description></description>
  <helper>
    <option value="image/gif">image/gif</option>
    <option value="image/jpg">image/jpg</option>
    <option value="image/png">image/png</option>
  </helper>
```



</element>

The recommended values are inside the section helper with the following format:

```
<option value="VALUE">LABEL_TO_SHOW_IN_EDITOR</option>
```

Sometimes an element is used in several sections, for example **bfs:propertyName** and is required to have either a different label or a different recommended values. This can be accomplished using the context attribute to define the XPath or parent element name.

An example for **bfs:propertyName** in OpenLayers section (with recommended values) and in TimeSeriesChart (without recommended values):

*<!-- OpenLayers property, with suggestions. Define the full xpath contexts to distinguish from time series and bar chart properties -->*

```
<element name="bfs:propertyName"
```

```
context="/bfs:MD_Metadata/bfs:layerInformation/bfs:MD_Layer/bfs:olProperty/bfs:MD_Property/bfs:propertyName">
```

```
<label>Name</label>
```

```
<description></description>
```

```
<helper>
```

```
<option value="buffer">buffer</option>
```

```
<option value="displayInLayerSwitcher">displayInLayerSwitcher</option>
```

```
<option value="isBaseLayer">isBaseLayer</option>
```

```
<option value="ratio">ratio</option>
```

```
<option value="singleTile">singleTile</option>
```

```
</helper>
```

```
</element>
```

*<!-- TimeSeriesChart property, without suggestions. Define the full xpath contexts to distinguish from time series and bar chart properties -->*

```
<element name="bfs:propertyName"
```

```
context="/bfs:MD_Metadata/bfs:layerInformation/bfs:MD_Layer/bfs:timeSeriesChartProperty  
/bfs:MD_Property/bfs:propertyName">
```

```
<label>Name</label>
```

```
<description></description>
```

```
</element>
```

Now you can rebuild the war again (see previous sections) and re-deploy the war.