

# **GeoNetwork Training**

***Release 3.4.2***

**GeoCat BV**

**June 29, 2018**



---

## Contents

---

<b>1</b>	<b>Modules</b>	<b>3</b>
1.1	Introduction to Metadata and Catalogues . . . . .	3
1.2	Configuring the catalog . . . . .	4
1.3	Profile Implementation . . . . .	7
1.4	Custom maps and layers . . . . .	11
1.5	Custom metadata views . . . . .	11
1.6	Custom Metadata Editor . . . . .	16
1.7	Integrate with CKAN . . . . .	29
1.8	Migrate from older versions of GeoNetwork . . . . .	29



Welcome to the GeoNetwork 3 data creator training manual by GeoCat. The manual contains hands-on exercises explaining how to install and configure the GeoNetwork system.

This course is an introduction to GeoNetwork 3 node usage from the perspective of a data creator.

Course modules focus on the use and deployment of GeoNetwork opensource from the getting started with metadata management, administration and production deployments.

This course is based on GeoNetwork 3.4. Newer versions of GeoNetwork are not guaranteed to contain the exact functionality as written, but should be reasonably similar.



## 1.1 Introduction to Metadata and Catalogues

These days the term metadata, also known as ‘data about data’, is used in a variety of contexts. Phone companies use the term for example to annotate phone conversations with additional tags, like which persons are calling for how long. In the spatial domain the term is generally used for documents that describe spatial assets, such as sensor observations, datasets, spatial services, maps, scientific articles and documents.

Metadata serves 2 main goals; discovery and assessment. Spatial catalogues, but also search engines, are able to ingest metadata and make it queryable, so users are able to discover resources that may be of interest to them. A second goal is assessment. Users are enabled to assess if a dataset is relevant for their use case by checking its metadata. Important attributes for assessment are potential usage or legal constraints (data license), date of modification, data quality and lineage information.

In recent decades the ISO Technical Committee 211 (ISO/TC211) together with Open Geospatial Consortium (OGC) has set up some standards for standardising metadata within the spatial domain. These standards are generally referred to as the ISO1911\* standards. Important examples are ISO19115, ISO19139, ISO19110, ISO19157.

On top of the encoding standards the standard “Catalogue Service for the web” (CSW) has been adopted, which is a standardised protocol to exchange and query metadata over the web. This standard facilitated the development of multiple catalogue clients which nowadays allow to query any spatial catalogue from a range of GIS clients, such as OpenLayers, ArcGIS and QGIS.

A typical Catalogue contains options to ‘harvest’ (ingest) metadata from other catalogues, in this way creating a network of catalogues facilitating wider discovery of spatial assets.

### 1.1.1 Introduction to GeoNetwork opensource

GeoNetwork opensource is a standard based and decentralised spatial information management system, designed to enable access to geo-referenced databases and cartographic products from a variety of data providers through descriptive metadata, enhancing the spatial information exchange and sharing between organisations and their audience, using the capacities and the power of the Internet.

The system provides a broad community of users with easy and timely access to available spatial data and thematic maps from multidisciplinary sources, that may in the end support informed decision making.

The main goal of the software is to increase collaboration within and between organisations for reducing duplication and enhancing information consistency and quality and to improve the accessibility of a wide variety of geographic information along with the associated information, organised and documented in a standard and consistent way.

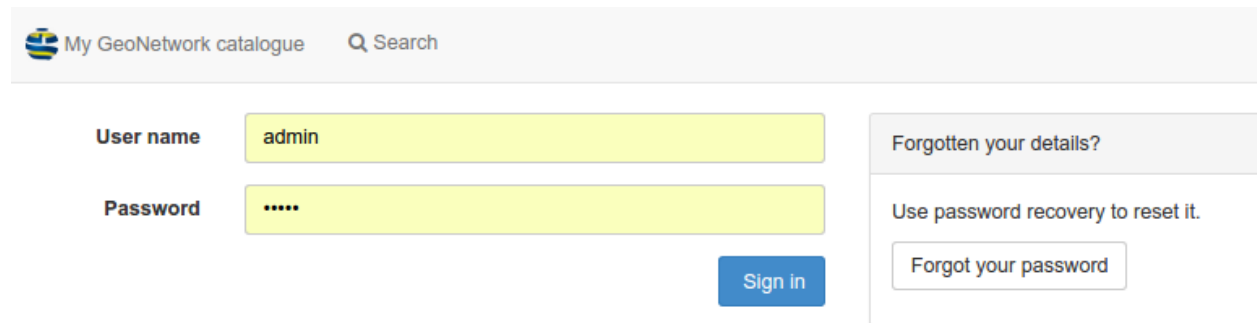
Main Features:

- Instant search on local and distributed geospatial catalogues
- Uploading and downloading of data, documents, PDF's and any other content
- An interactive Web map viewer that combines Web Map Services from distributed servers around the world
- Online map layout generation and export in PDF format
- Online editing of metadata with a powerful template system, including validation options
- Scheduled harvesting and synchronisation of metadata between distributed catalogues
- Groups and users management
- Fine grained access control

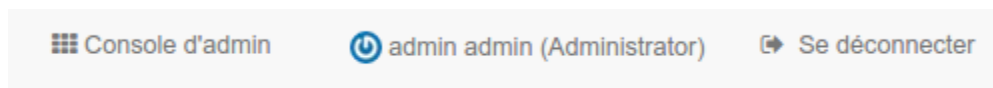
## 1.2 Configuring the catalog

This section describes the minimal configuration for the catalog. In the administration training are described all the settings in detail.

Click the `sign in` to connect as administrator. The default account is username `admin` with password `admin`.



Once connected, the top toolbar should provide a link to the `Admin console` and your login details.



Most of the system configuration parameters can be changed by administrator users using the web interface in `Admin console > Settings`.

---

**Important:** Configuration of these parameters is critically important for the catalog in an operational context. Misunderstanding some settings may result in a system that does not function as expected. For example, downloads may fail to be correctly processed, or metadata harvesting from other servers may not work.

---



Settings
Logo
Sources
CSW
Virtual CSW
CSW test
Map servers

Settings
Production
Save settings

## System settings

### Catalog description

Catalog name	My GeoNetwork catalogue
	The name displayed in different places (eg. news feed).
Catalogue identifier	ba71fb75-a66f-4f4a-bc4b-20f3994eb9d6
	Unique catalogue identifier. After changing the site identifier, user should update the index and the logo from the administration panel.
Organization	My organization
SVN UUID	43487abf-8957-4df6-99b1-9cf8d99c7781
	Subversion repository unique identifier

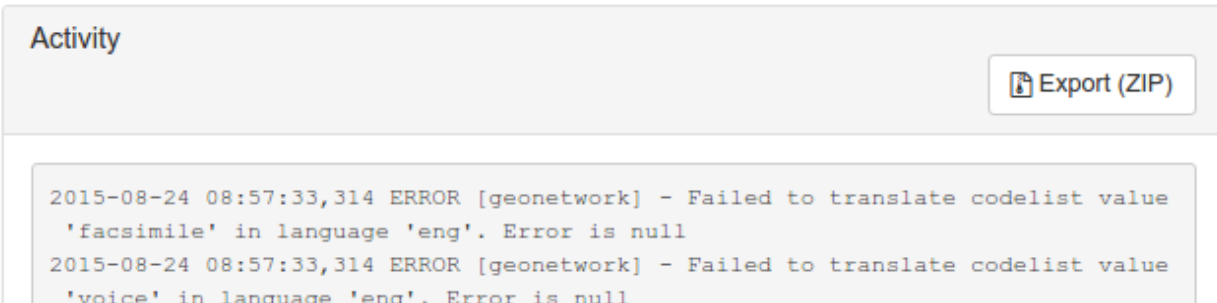
### 1.2.1 Catalog description

- **Catalog came** The name of the node. Information that helps identify the catalogue to a human user. The name is displayed on the banner, in the CSW GetCapabilities.
- **Catalog identifier** A universally unique identifier (uuid) that distinguishes your catalog from any other catalog. This is a unique identifier for your catalogue and its best to leave it as a uuid. It will be used by harvester using GeoNetwork protocol to identify the source catalog.
- **Organization** The organization the node belongs to. Again, this is information that helps identify the catalogue to a human user.
- **SVN UUID** Subversion repository attached to the node. This identifier is created and/or checked on startup to verify that the database matches the SVN repository. The repository is used for metadata versioning.

### 1.2.2 Catalog Server

- **Host** The node's name or IP number (without <http://>). For example, they are used during metadata editing to create resource links and when returning the server's capabilities during a CSW request.
- If your node is publicly accessible from the Internet, you have to use the domain name.
- If your node is hidden inside your private network and you have a firewall or web server that redirects incoming requests to the node, you have to enter the public address of the firewall or web server. A typical configuration is to have an Apache web server on address A that is publicly accessible and redirects the requests to a Tomcat server on a private address B. In this case you have to enter A in the host parameter.
- **Port** The server's port number (usually 80 or 8080). If using HTTP, set it to 80.
- **Preferred Protocol** Defined the protocol to access the catalog. The HTTP protocol used to access the server. Choosing http means that all communication with the catalog will be visible to anyone listening to the protocol. Since this includes usernames and passwords this is not secure. Choosing https means that all communication with the catalog will be encrypted and thus much harder for a listener to decode.
- **Secure Port** The secure port number.

- **Log level** Define the logging level of the application. After modification, log can be checked in the Statistics & status section under Activity.



### 1.2.3 Intranet parameters

A common need for an organisation is to automatically discriminate between anonymous internal users that access the node from within an organisation (Intranet) and anonymous external users from the Internet. The catalog defines anonymous users from inside the organisation as belonging to the group *Intranet*, while anonymous users from outside the organisation are defined by the group *All*. To automatically distinguish users that belong to the Intranet group you need to tell the catalog the intranet IP address and netmask.

- **Netmask** The intranet netmask (eg. 255.255.255.0).
- **Network\*** The intranet address in IP form (eg. 147.109.100.0).

### 1.2.4 Proxy server

- **Use proxy** Enable the proxy in case the catalog is behind a proxy and need to use it to access remote resources.
- **Proxy Host** The proxy IP address or name
- **Port** The proxy port
- **Proxy username** The username
- **Proxy user password** The username password
- **\*Ignore host list** To bypass specific hosts enter a specific IP address or host name such as www.mydomain.com or an address range using wildcards, such as 192.168.2.\*. Use | to separate the different host values.

JVM proxy parameters may also be required to properly set the proxy for all remote access.

### 1.2.5 Feedback

Email may be sent by the catalog.

- you are using the User Self-registration system
- you are using the metadata status workflow
- a file uploaded with a metadata record is downloaded and notify privilege is selected

This section configure the mail server to use.

- **Email** This is the administrator's email address used to send feedback.
- **SMTP host** The mail server name or IP address to use for sending emails.

- **SMTP port** The SMTP port.
- **Use SSL** Enable SSL mode
- **User name** Username if connection is required on the SMTP server
- **Password** Username password if connection is required on the SMTP server

## 1.3 Profile Implementation

Although there are many schema plugins available, sometimes we need to adapt the XSD definition or the validation rules to our use cases. On this section we will learn how to do it on a clean easy way.

### 1.3.1 Create a new profile

To create a new profile you should start by selecting an already existing schema plugin. Creating a full schema plugin from scratch is out of scope of this manual.

#### Choose the schema plugin base

The best way to find a schema plugin is visiting the Metadata 101 repository in <https://github.com/Metadata101>

On that repository, we can find both base ISO schemas and some of the profiles already used by some organizations. We choose the schema that adapts better to our needs and clone it on the schemas folder of GeoNetwork.

On this case, we are going to use ISO19139 to generate ISO19139-custom:

```
$ cd core-geonetwork
$ cp -r schemas/iso19139/ schemas/iso19139-custom
$ ls schemas/iso19139-custom
  doc  pom.xml  src  target
$ mv schemas/iso19139-custom/src/main/plugin/iso19139/ schemas/iso19139-custom/src/
↪main/plugin/iso19139-custom
```

Now we have to make sure the schema is copied on the right repository. To do this we add the following resource in pom.xml:


```
<resource>
  <directory>${project.basedir}/../schemas/iso19139-custom/src/main/plugin</directory>
  <targetPath>${basedir}/src/main/webapp/WEB-INF/data/config/schema_plugins</
↪targetPath>
</resource>
```

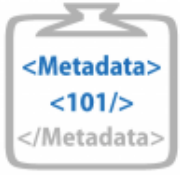
This will make sure the files are correctly copied on the schema plugins folder

#### Adapt the XSD

Even when we are using an ISO schema, sometimes we may want to tweak the definition of that schema to adapt to our needs.

The xsd is defined on the schemas/iso19139-custom/src/main/plugin/iso19139/schema folder. Different namespaces are defined in different folders, so you will have to search for the element you want to modify on this folder and define the XSD rules that apply to that element. For example, we may want to restrict the options available for a conditional element.

 This organization Search Pull requests Issues Marketplace Explo



# Metadata 101

<http://www.metadata101.org>

Repositories 13

People 10

Teams 2

Projects 0

Settings

Search repositories... Type: All Language: All

## iso19115-3

XSLT ★ 2 7 Updated 18 hours ago

## iso19139.nl.services.1.2.1

XSLT 1 Updated 8 days ago

## iso19139.nl.geografie.1.3.1

XSLT 1 Updated 9 days ago

## iso19139.mcp-2.0

ISO19115/19139 Marine Community Profile Version 2.0

XSLT 2 Updated 9 days ago

## iso19139.anzlic

ISO19115/19139 ANZLIC Profile 1.1

XSLT 1 Updated 9 days ago

On the following xml extracted from schema/gmd/identification.xsd, we can force a minOccurs 1 in some of the elements that explicitly defines minOccurs as 0. This way we will force the XSD rules to check that there is a minimum of one element of that type on the identification gmd definition.

```
<xs:complexType name="AbstractMD_Identification_Type" abstract="true">
  <xs:annotation>
    <xs:documentation>Basic information about data</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="gco:AbstractObject_Type">
      <xs:sequence>
        <xs:element name="citation" type="gmd:CI_Citation_PropertyType"/>
        <xs:element name="abstract" type="gco:CharacterString_PropertyType"/>
        <xs:element name="purpose" type="gco:CharacterString_PropertyType" minOccurs=
↪ "0" />
        <xs:element name="credit" type="gco:CharacterString_PropertyType" minOccurs="0
↪ "
          maxOccurs="unbounded"/>
        <xs:element name="status" type="gmd:MD_ProgressCode_PropertyType" minOccurs="0
↪ "
          maxOccurs="unbounded"/>
        <xs:element name="pointOfContact" type="gmd:CI_ResponsibleParty_PropertyType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="resourceMaintenance" type="gmd:MD_MaintenanceInformation_
↪ PropertyType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="graphicOverview" type="gmd:MD_BrowseGraphic_PropertyType" ↪
↪ minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element name="resourceFormat" type="gmd:MD_Format_PropertyType" minOccurs=
↪ "0"
          maxOccurs="unbounded"/>
        <xs:element name="descriptiveKeywords" type="gmd:MD_Keywords_PropertyType" ↪
↪ minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element name="resourceSpecificUsage" type="gmd:MD_Usage_PropertyType" ↪
↪ minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:element name="resourceConstraints" type="gmd:MD_Constraints_PropertyType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="aggregationInfo" type="gmd:MD_AggregateInformation_
↪ PropertyType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Although it is advisable to follow the ISO rules and only make the XSD more strict to still be compliant with the standards, GeoNetwork does not force you to follow a standard. You can create any set of XSD rules and that's what GeoNetwork will use to show, edit and validate your metadata.

### Adapt the validation rules

It is very common to adapt the validation rules to make sure your data complies with your specific needs.

There are two types of validation rules. The most basic set of rules is based on the same XSD we have been modifying on the previous section. GeoNetwork just makes sure that the xml of the metadata is conformant with the XSD

definition.

But for more complex rules we have the schematrons. Schematrons allow you to define rules based on the content or conditional rules that depend on other elements. Schematrons are files placed on a folder called “schematrons”. You can have disabled schematrons if you add “disabled” to the filename.

A schematron pattern contains a title and a rule. The rule can make some checks on elements and if the test does not pass, it will generate an (internationalized) error message to describe the problem, which will be shown on the metadata editor validation box.

On the following example, this schematron checks values of density inside the gmd:MD\_Medium element: if there is a value for density, it has to have a defined unit for density:

```
<sch:pattern>
  <sch:title>$loc/strings/M18</sch:title>
  <sch:rule context="//gmd:MD_Medium">
    <sch:let name="density" value="gmd:density and
    ↪not (gmd:densityUnits[@gco:nilReason!='missing' or not (@gco:nilReason)])"/>
    <sch:assert test="$density = false()"
      >$loc/strings/alert.M18</sch:assert>
    <sch:report test="$density = false()"
      ><sch:value-of select="$loc/strings/report.M18"/> <sch:value-
    ↪of select="gmd:density"/>
      <sch:value-of select="gmd:densityUnits/gco:CharacterString"/>
    ↪</sch:report>
  </sch:rule>
</sch:pattern>
```

We can add as many patterns as we want to the schematron file.

Sometimes we want to apply a certain set of rules only if some condition apply. For example, we may want to reinforce some specific rules for all metadata that contains a specific keyword. We may add that conditional to each pattern of the schematron file or configure on the admin console what schematron set of rules will be applied on each case:

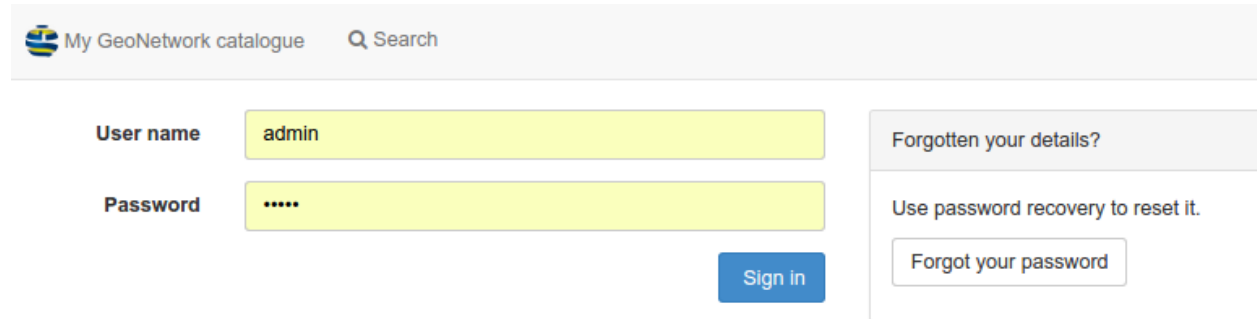
geonetwork/srv/eng/admin.console#/metadata/schematron/iso19139/108

The screenshot displays the GeoNetwork Admin Console interface for configuring schematron rules. On the left, a sidebar titled 'Schematrons' shows a tree structure under 'iso19139', including 'Geonetwork Rules', 'INSPIRE SDS Rules' (selected), 'INSPIRE Strict Rules', 'INSPIRE Rules', 'ISO Rules', and 'URL Validation'. The main content area is titled 'Schematron Applicability'. It features a search bar with '\*Generated\*' and a '+', a dropdown for 'Required to be valid', and a criteria box stating 'This criteria always passes'. Below the criteria box is a 'Keyword' dropdown and 'Save'/'Cancel' buttons.

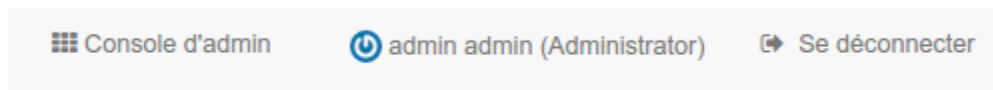
### 1.3.2 Load the created custom profile

Now we can try our custom iso19139 schema and use it.

Click the `sign in` to connect as administrator.



Once connected, the top toolbar should provide a link to the `Admin console` and your login details.



Go to the `admin console` and click on `metadata and templates`:

Choose all standards and click `load samples` and then `load templates` in order to load examples:

Go back to the search page to see examples:

## 1.4 Custom maps and layers

### 1.4.1 Background layers

### 1.4.2 Store a map




GeoNetwork allows you to create an `OWSContext` to store a map and load it later.


Although the main feature of GeoNetwork is not store data, you can store this `OWSContext` on GeoNetwork and serve it with the proper metadata.


## 1.5 Custom metadata views


GeoNetwork facilitates developers to easily change or add a metadata view. A User can change the view to his/her needs.


By default the initial view is an AngularJS view on the results returned from the search service. Therefore the view can only contain the fields from the Lucene Index. If you require more fields, you can either add fields to the index







 My GeoNetwork catalogue  Search  Map

  
Metadata and templates

  
Harvesting

  
Reports

  
Classification system

 My GeoNetwork catalogue  Search  Map  Contribute  Admin console  admin ac

Metadata & templates

Formatter

Schematron

Metadata Identifier templates

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

Load templates for selected standards

Load samples for selected standards

12

Chapter 1. Modules



**My GeoNetwork catalogue**    🔍 Search    🗺 Map    ➕ Contribute    ⚙ Admin console    👤 admin admin (Administrator)    ✎ Sign out    English ▼

---

🔍 ×

☐ -

**TYPE OF RESOURCES**

- Dataset (3)
- Maps and graphics (1)
- Collection session (1)
- Service (1)

**TOPICS**

- Geoscientific information (1)
- Boundaries (1)
- Inland waters (1)

**KEYWORDS**

- Polar ecosystem (1)
- Physiography, soil (1)
- Eurasia (1)
- Geoscientific information (1)

BOUNDARIES-Administrative (1)

[10 more](#)

**CONTACT FOR THE RESOURCE**

- FAO - Land and Water Development Division (1)
- Department of Sustainability and Environment (DSE) (1)

**YEARS**


- 2010 (1)
- 2007 (1)
- 2000 (1)

**FORMATS**

- Web page (1)

Sorted by relevancy • 1 - 6 on 6 • < >


Categories [icon]



### Hydrological Basins in Africa (Sample record, please remove!)

[edit] [share] [comment]


Categories [icon]



### Geoscience Australia's Open Day Photographs 26th August 2007

[edit]


Categories [icon]



### The Geoffrey's Tube Z3950 Server (Sample Record - Please Delete!)

[edit]

Categories [icon]




### Localities in Victoria (VMADMIN.LOCALITY\_PO LYGON) - Comprehensive

Department of Sustainability and Environment (DSE)

[edit]

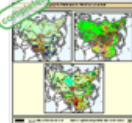
Categories [icon]



### Natural polar ecosystems

[edit]

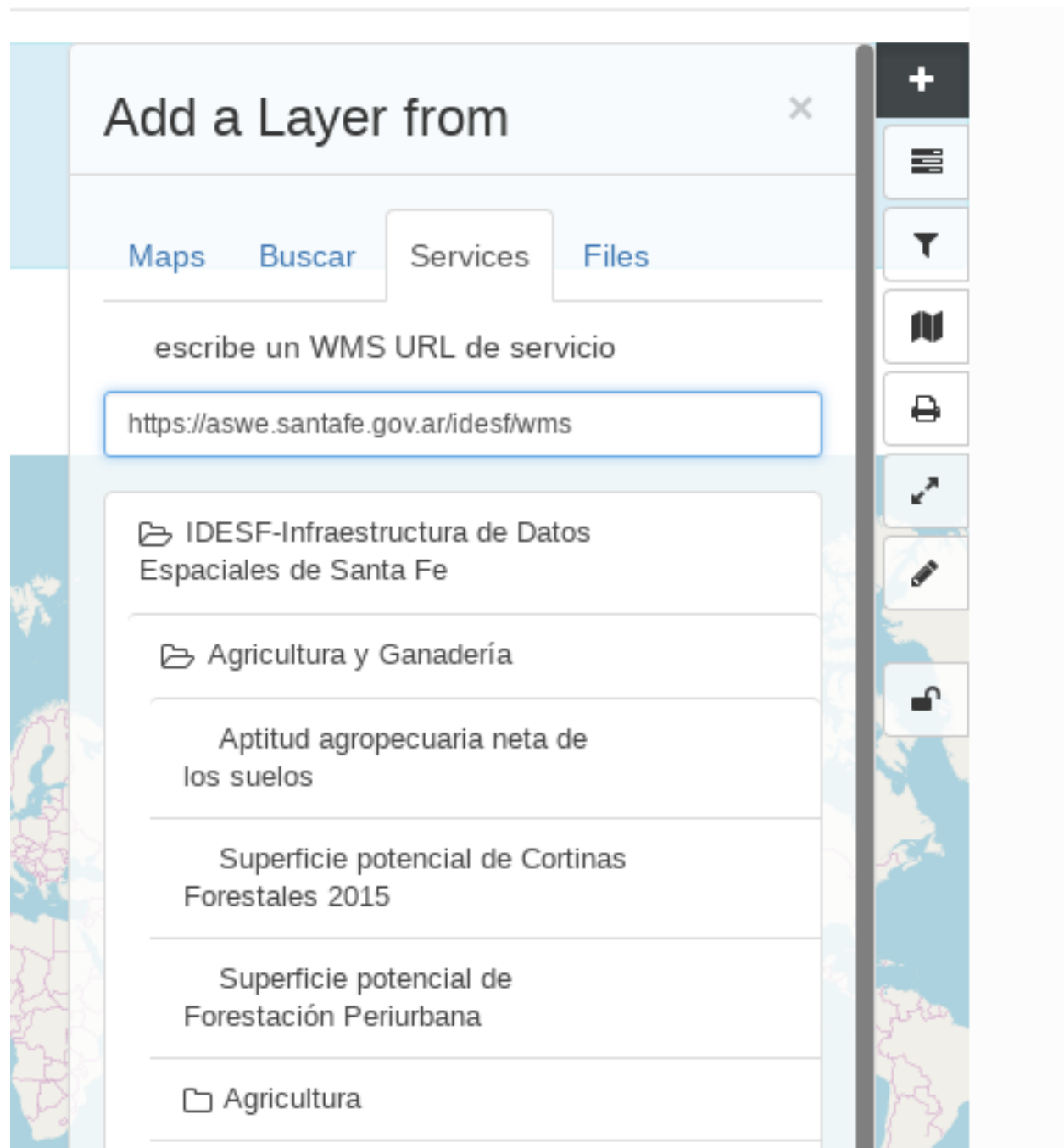
Categories [icon]

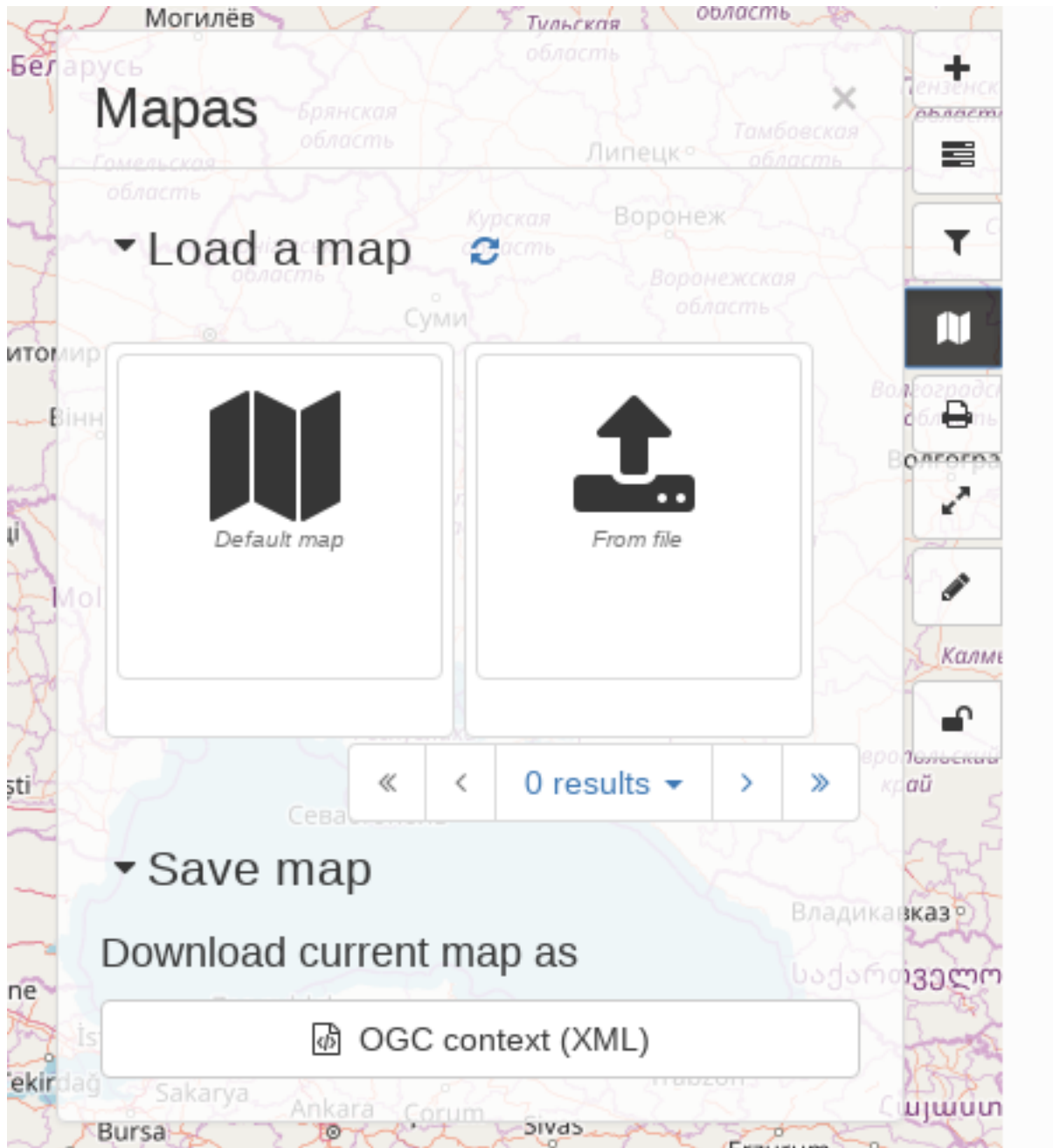


### Physio North and (Sample

FAO - Land and W

[edit] [share] [comment]





or don't use the AngularJS view. This view is defined in `web-ui/src/main/resources/catalog/views/default/templates/recordView.html`.

Metadata views are called 'formatters'. They are located in the schema-plugin related to the metadata that you are formatting. Formatters use either XSLT or Groovy to transform the XML to the required format (html, xml, pdf, json).

A formatter can be updated from the web interface in the admin console, metadata and templates, tab 'formatters'. On this page you can upload, change and preview formatters.

The screenshot displays the 'Formatters' management interface. At the top, there are three tabs: 'Metadata & templates', 'Formatter' (selected), and 'Schematron'. The 'Formatter' tab is divided into two main sections. On the left, a 'Formatter' list shows a table with columns for 'Filter' and 'Formatter'. The table contains five entries: '/xml\_view', 'iso19139/full\_view', 'csw-record/full\_view', 'dublin-core/full\_view', and 'iso19110/xsl-view'. Below this table is an 'Add a formatter' section featuring a green button labeled '+ Choose a formatter package to upload' and a text box explaining that a formatter is a ZIP file containing metadata layout definitions. On the right side of the page, there are two configuration panels. The top panel, 'Configure formatter /xml\_view', includes a 'Choose a file' dropdown, a text area for content, a 'Save file' button, and a 'Delete' button. The bottom panel, 'Test formatter /xml\_view', has a 'Choose a metadata internal identifier' dropdown and three buttons: 'View XML', 'View HTML', and 'View debug'.

After having created a new formatter you will have to update your application code, so the new formatter-output can be visualised from your application. If the goal of the formatter is to introduce a new html view on metadata, then you can add a reference to it in `web-ui/src/main/resources/catalog/views/default/config.js` (`searchSettings.formatter.list`).

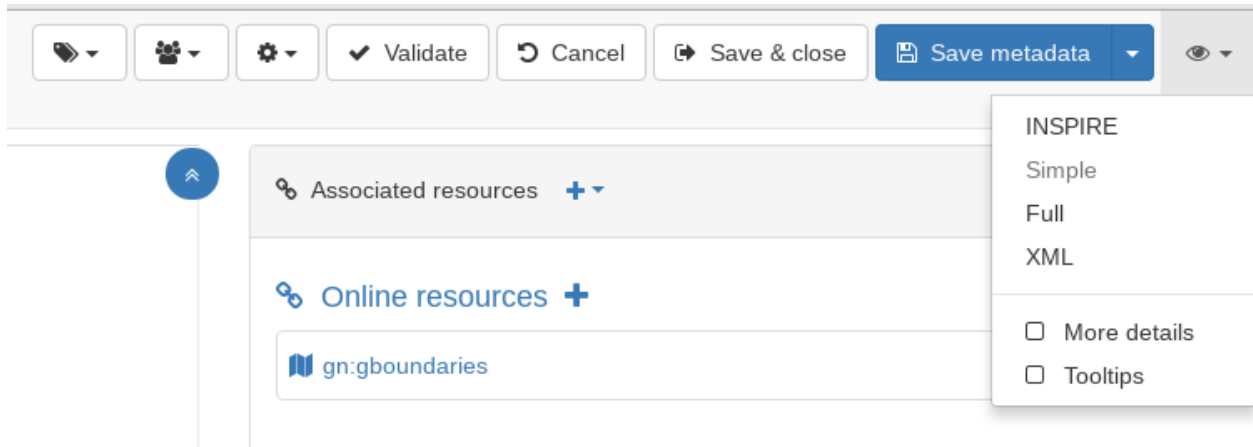
## 1.6 Custom Metadata Editor

GeoNetwork helps data creators to easily change or add a metadata editor view. The metadata editor views are specific for each schema, so don't forget to update all your schemas. Usually a schema has three different metadata views defined: Simple, Full and XML view. If you have INSPIRE enabled, usually another INSPIRE view is enabled on your editor.

The simple view contains the most common elements edited. This is usually helpful for data creators that may not be interested on the full schema.

The full view is generated based on the XSD schema and contains all the possible elements available on that schema. This advanced view is useful for complex content creation, as it allows the editor to add, remove and fill any possible combination of elements (according to the XSD).

The xml view is a simple colored view that displays the xml of the metadata in a tree mode. This view should be used only for very advanced users, as it is easy to break the xml and make it not compliant with XSD. It is useful for users



that may want to skip validation rules.

Validation rules are the same for all the views. The views only define what elements are shown on the editor. If some element is not shown on the editor, that doesn't mean it is being removed from the metadata. An element not shown on an editor view will not be modified when saving the metadata.

You can create as many views as you want and you may want to create a different view depending on the profile of your data creator.

This views are defined on an xml file called config-editor.xml inside the layout folder.

To build such an editor configuration user needs to know the XSD of the standard to properly build views, tabs and fields according to element names (see `schemas/config-editor.xsd`). Create an editor root element and attach:

- the schema and
- namespaces for the standards

```
<editor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../../../config-editor.xsd"
  xmlns:gn="http://www.fao.org/geonetwork"
  xmlns:gco="http://www.isotc211.org/2005/gco"
  xmlns:gmd="http://www.isotc211.org/2005/gmd"
  xmlns:gmx="http://www.isotc211.org/2005/gmx"
  xmlns:srv="http://www.isotc211.org/2005/srv"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink">
```

An editor configuration should define first some general element description and then a set of views with at least one.

Child elements:

- **fields**, Optional element (see creating-custom-editor-fields)
- **fieldsWithFieldset**, Optional element (see creating-custom-editor-fieldsWithFieldset)
- **multilingualFields**, Optional element (see creating-custom-editor-multilingualFields)
- **views**, Mandatory element (see creating-custom-editor-views)

### 1.6.1 Fields

```
<!-- Form field type configuration. Default is text. -->
<fields>
  <for name="gmd:abstract" use="textarea"/>
  <for name="gco:Distance" use="number"/>
  <for name="gmd:onLine" addDirective="data-gn-directory-entry-selector">
    <directiveAttributes data-template-add-action="true" data-template-type="onLine"
    ↪data-filter='{ "_root": "gmd:CI_OnlineResource"}' />
  </for>
</fields>
```

This defines for each XSD defined type, which type of input to use. As with `gmd:onLine`, a customized angular input type can be defined.

You can define the form fields type configuration. Default is simple text input. This list contains the list of exception which does not use a simple text input.

- all HTML5 input type or
- an AngularJS directive name.

An element can only have one type defined.

```
<editor>
  <fields>
    <for name="gmd:abstract" use="textarea"/>
    <for name="gco:Real" use="number"/>
    <for name="gco:Boolean" use="checkbox"/>
    <for name="gco:Date" use="data-gn-date-picker"/>
  </fields>
</editor>
```

The other option to define a more advanced field type is to catch the element using and XSL template. This approach is used for keywords in ISO19139 for example (see `schemas/iso19139/src/main/plugin/iso19139/layout/layout-custom-fields-keywords.xsl`).

### 1.6.2 Fields with Fieldset

After defining all the simple fields, we define a list of fieldsets, which the editor will recognize as something to put together, a list of element to be displayed in a fieldset (ie. boxed element). Those elements usually contain children elements and define major section in the standard. For example, in ISO19139, identification and distribution are major section and should usually be displayed as a group of information.

```
<editor>
  <fields>...</fields>
  <fieldsWithFieldset>
    <name>gmd:identificationInfo</name>
    <name>gmd:distributionInfo</name>
  </fieldsWithFieldset>
</editor>
```

### 1.6.3 Multilingual Fields

Now we define what fields are multilingual. By default, if the standard as multilingual support like ISO19139, all fields will be displayed as multilingual fields. Define in the exclude section the exception (eg. `gmd:identifier` for example in ISO19139).

Then this section also allows to define how multilingual fields are displayed using the expanded elements. If expanded, then one field per language is displayed with no need to click on the language switcher.

```

<multilingualFields>
  <!-- In multilingual mode, define which mode the widget should have. If expanded,
  then one field per language is displayed. -->
  <expanded>
    <name>gmd:title</name>
    <name>gmd:abstract</name>
  </expanded>
  <exclude>
    <name>gmd:identifier</name>
  </exclude>
</multilingualFields>

```

## 1.6.4 Views

Finally, we define the views, which will customize which fields will be shown when editing the metadata. We can define more than one view per schema, so the editor user can select which one to use when editing the metadata.

At least one view **MUST** be defined but more view modes can be defined depending on the needs.

When a non-final field is defined on a view, the editor will automatically generate all the input fields inside it. So, you don't have to explicitly define all the fields you want to show. The simplest view is the one based on the XSD of the schema.

The xml view can also be easily defined:

```

<view name="xml">
  <tab id="xml" default="true"/>
</view>

```

If the view is too big, we can also define a set of tabs inside the editor, so not all fields are shown at the same time. Remember that when the user changes the tab, the content will be saved. So we should place related fields inside the same tab.

## 1.6.5 Defining a view

A view has a label and define a specific rendering of the metadata records. A view is composed of one or more tabs.

```

<views>
  <view name="custom-view">
    ...
  </view>

```

The view could be displayed or not according to the metadata record content or the current user session using the `displayIfRecord` and `displayIfServiceInfo` attribute.

Attributes:

- **name** (Mandatory)

The key of the view name stored in `{schema}/loc/{lang}/strings.xml` or the element name with namespace prefix.

```

<strings>
  <default>Simple</default>
  <inspire>INSPIRE</inspire>
  <custom-view>My view</custom-view>

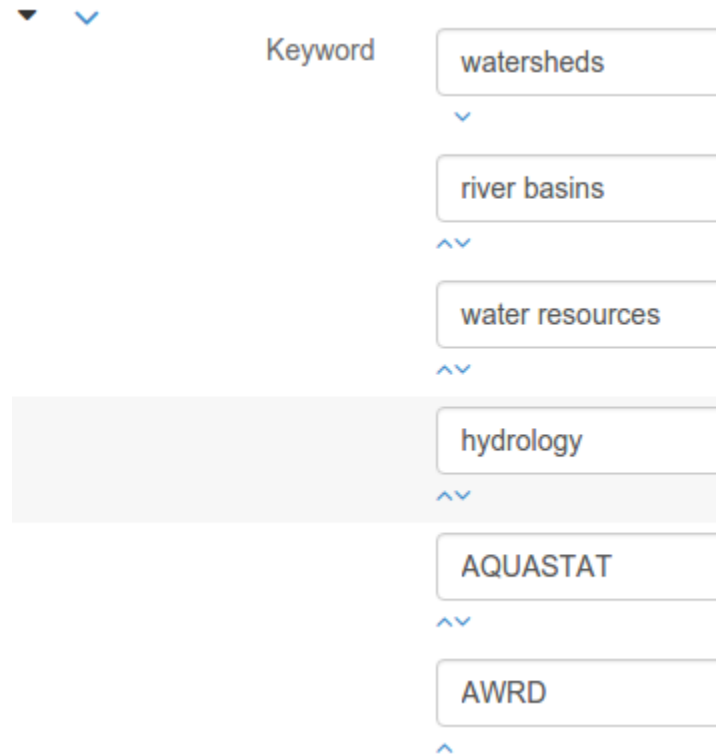
```

- **disabled** (Optional) Fixed value: **true**

Hide the view from the menu if the attribute is defined. Allows to easily disable a view.

- **upAndDownControlHidden** (Optional) Fixed value: **true**

Define if up and down control should be displayed in that view. If not defined, controls are displayed. Hide those controls in a view to make it easier with less controls for the end-user.



- **displayIfRecord** (Optional)

XPath expression returning boolean value which will be evaluated against the metadata record. if true the view will be displayed. eg. Display custom-view if metadata standard name contains Medsea:

```
<view name="custom-view"
      displayIfRecord="contains(gmd:MD_Metadata/
                              gmd:metadataStandardName/gco:CharacterString,
                              'MedSea') "
```

- **displayIfServiceInfo** (Optional)

XPath expression returning boolean value which will be evaluate against the service information tree (Jeeves /root/gui element). if true the view will be displayed.

eg. Display custom view if user is Administrator:

```
<view name="custom-view"
      displayIfServiceInfo="count(session[profile = 'Administrator']) = 1"
```

displayIfRecord and displayIfServiceInfo could be combined. An AND operator is used. Both condition MUST returned true for the view to be displayed.

- **hideTimeInCalendar** (Optional) Fixed value: **true**



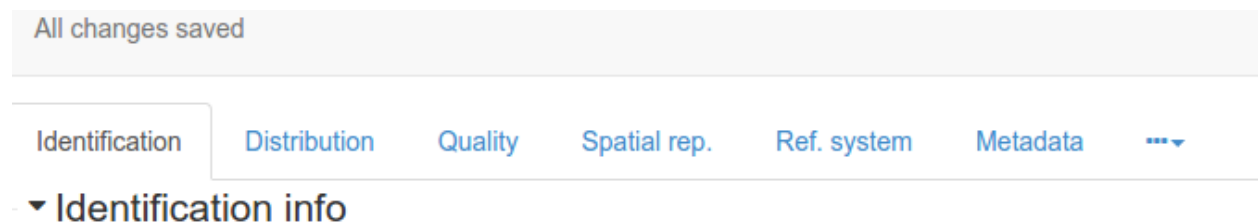
Define if calendar control should allow users to set date only or datetime. If attribute is not set, then date and time can be set. This is controlled at the view level, switching to another view may allow more control over the dates.

Child elements:

- **tab**
- **flatModeExceptions**
- **thesaurusList**

## 1.6.6 Defining a tab

A view contains at least one tab. In that case it will be the default to display and no top toolbar will be displayed to switch from one tab to another.



Add custom view one default tab and a field for the title:

```
<views>
  <view name="custom-view">
    <tab id="custom-tab" default="true">
      <section>
        <field xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/*/
↪gmd:title"/>
      </section>
    </tab>
  </view>
```

## Configuring complex element display

Elements to apply “flat” mode exceptions. By default, “flat” mode does not display elements containing only children and no value.

```
<views>
  <view name="custom-view">
    <tab id="custom-tab" default="true">
      <section>
        <field xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/*/
↪gmd:title"/>
        <field name="pointOfContact"
          xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:pointOfContact"
          del=". "/>
      </section>
    </tab>
  </view>
```

eg. To display gmd:descriptiveKeywords element even if does not exist in the metadata record or if the field should be displayed to add new occurrences:

```
</tab>

<!-- Elements that should not use the "flat" mode -->
<flatModeExceptions>
  <for name="gmd:descriptiveKeywords" />
</flatModeExceptions>
</view>
```

## Customizing thesaurus

To configure the type of transformations or the number of keyword allowed for e thesaurus define a specific configuration:

eg. only 2 INSPIRE themes.

```
<thesaurusList>
  <thesaurus key="external.theme.inspire-theme"
    maxtags="2"
    transformations="" />
</thesaurusList>
```

### 1.6.7 Adding a section to a tab

A section is a group of fields.

Attributes:

- **name** (Optional)

An optional name to override the default one base on field name for the section. The name must be defined in {schema}/loc/{lang}/strings.xml.

- **xpath** (Optional)

The xpath of the element to match. If an XPath is set for a section, it should not contains any field.

- **mode** (Optional) Fixed value: **flat**

The “flat” mode is an important concept to understand for the editor. It controls the way:

- complex elements are displayed (ie. elements having children) and
- non existing elements are displayed (ie. elements in the standard not in the current document).

When a tab is in flat mode, this tab will not display element which are not in the current metadata document and it will display complex element as a group only if defined in the list of element with fieldset (see creating-custom-editor-fieldsWithFieldset).

Example for a contact in non “flat” mode:

Example for a contact in “flat” mode:

This mode makes the layout simpler but does not provide all controls to remove some of the usually boxed element. End-user can still move to the advanced view mode to access those hidden elements in flat mode.

It's recommended to preserve at least one view in non “flat” mode for reviewer or administrator in order to be able:

- to build proper templates based on the standards
- to fix any types of errors.

▼ Point of contact

▼ Responsible party

Individual name

Organisation name

Position name

▼ Contact Information

▼ Contact

Phone

▼ Address

▼ Address

Delivery point

City

Administrative area

Postal code

Country

Electronic mail address

▼ Point of contact

Organisation name

▼ Address

Electronic mail address

Role

- **or** (Optional)

Local name to match if the element does not exist.

- **or** (Optional)

The local name of the geonet child (ie. non existing element) to match.

```
<field xpath="/gmd:MD_Metadata/gmd:language"
      or="language"
      in="/gmd:MD_Metadata"/>
```

- **in** (Optional)

XPath of the geonet:child element with the or name to look for. Usually points to the parent of last element of the XPath attribute.

- **in** (Optional)

The element to search in for the geonet child.

### 1.6.8 Adding a field

To display a simple element use the `xpath` attribute to point to the element to display:

```
<field xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/*/gmd:title"/>
```

To override a field label use the `name` attribute and define that new label in `{schema}/loc/{lang}/strings.xml`:

```
<field name="myTitle"
      xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/*/gmd:title"/>
```

To display a complex element which exist in the metadata document:

```
<field name="pointOfContact"
      xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:pointOfContact"/>
```

In that case all children elements are also displayed.

To display a field if exist in the metadata document or providing a add button in case it does not exist (specify `in` and `or` attribute):

```
<field name="pointOfContact"
      xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:pointOfContact"
      or="pointOfContact"
      in="/gmd:MD_Metadata/gmd:identificationInfo/*"
      del="."/>
```

Activate the “flat” mode at the tab level to make the form display only existing elements:

```
<view name="custom-view">
  <tab id="custom-tab" default="true" mode="flat">
    <section>
      <field
        xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/*/
↪gmd:title"/>
      <field name="pointOfContact"
        xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/gmd:pointOfContact"
```

(continues on next page)

(continued from previous page)

```

        or="pointOfContact"
        in="/gmd:MD_Metadata/gmd:identificationInfo/*"
        del="."/ >
    </section>
</tab>
</view>

```

Attributes:

- **xpath** (Mandatory)

The xpath of the element to match.

- **if** (Optional)

An optional xpath expression to evaluate to define if the element should be displayed only in some situation (eg. only for service metadata records). eg.

```

<field
  xpath="/gmd:MD_Metadata/gmd:identificationInfo/srv:SV_ServiceIdentification/
gmd:resourceConstraints/gmd:MD_LegalConstraints/gmd:otherConstraints"
  if="count (gmd:MD_Metadata/gmd:identificationInfo/srv:SV_ServiceIdentification) > 0"/
  >

```

- **name** (Optional)

A field name to override the default name.

- **isMissingLabel** (Optional)

The label to display if the element does not exist in the metadata record. It indicates that the element is missing in the current record. It could be use for a conformity section saying that the element is “not evaluated”. EXPERIMENTAL

- **or** (Optional)

The local name of the geonet child (ie. non existing element) to match.

```

<field xpath="/gmd:MD_Metadata/gmd:language"
  or="language"
  in="/gmd:MD_Metadata"/>

```

- **in** (Optional)

The element to search in for the geonet child.

- **del** (Optional)

Relative XPath of the element to remove when the remove button is clicked.

eg. If a template field match linkage and allows editing of field URL, the remove control should remove the parent element gmd:onLine.

```

<field name="url"
  xpath="/gmd:MD_Metadata/gmd:distributionInfo/gmd:MD_Distribution/gmd:transferOptions
  /gmd:MD_DigitalTransferOptions/gmd:onLine/gmd:CI_OnlineResource/
  >gmd:linkage"
  del="../../..">
<template>

```

- **templateModeOnly** (Optional) Fixed value: **true**

Define if the template mode should be the only mode used. In that case, the field is always displayed based on the XML template snippet field configuration. Default is false.

- **notDisplayedIfMissing** (Optional) Fixed value: **true**

If the field is found and a geonet child also, the geonet child to add a new one is not displayed.

Child elements:

- **template**, Optional element (see *Adding a template based field*)

### 1.6.9 Adding a template based field

A template configuration for an XML snippet to edit.

A template field is composed of an XML snippet corresponding to the element to edit where values to be edited are identified using `{{fields}}` notation. Each field needs to be defined as values from which one input field will be created.

This mode is used to hide the complexity of the XML element to edit. eg.

```
<field name="url"
  templateModeOnly="true"
  xpath="/gmd:MD_Metadata/gmd:distributionInfo/g.../gmd:linkage">
  <template>
    <values>
      <key label="url"
        xpath="gmd:URL"
        tooltip="gmd:linkage"/>
    </values>
    <snippet>t
      <gmd:linkage>
        <gmd:URL>{{url}}</gmd:URL>
      </gmd:linkage>
    </snippet>
  </template>
```

The template field mode will only provide editing of part of the snippet element. In some cases the snippet may contain more elements than the one edited. In such cases, the snippet **MUST** identify the list of potential elements in order to not lose information when using this mode. Use the `gn:copy` element to properly combine the template with the current document.

eg. The `gmd:MD_Identifier` may contain a `gmd:authority` node which needs to be preserved.

```
<snippet>
  <gmd:identifier>
    <gmd:MD_Identifier>
      <gn:copy select="gmd:authority"/>
      <gmd:code>
        <gco:CharacterString>{{code}}</gco:CharacterString>
      </gmd:code>
    </gmd:MD_Identifier>
  </gmd:identifier>
</snippet>
```

### 1.6.10 Adding documentation or help

Insert an HTML fragment in the editor.

```
<field name="edmerpName"
  xpath="/gmd:MD_Metadata/gmd:identificationInfo/*/
    gmd:pointOfContact[*/gmd:role/*/codeListValue='edmerp']"
  del=".">

<text ref="edmerp-help"/>
```

The fragment is defined in localization file strings.xml:

```
<edmerp-help>
  <div class="row">
    <div class="col-xs-offset-2 col-xs-8">
      <p class="help-block">The European Directory for Marine Environment
        Research Project (EDMERP) contains descriptions of many projects.
        This catalogue is maintained ...</p>
    </div>
  </div>
</edmerp-help>
```

Attributes:

- **ref** (Mandatory)

The tag name of the element to insert in the localization file.

### 1.6.11 Adding a button

A button which trigger an action (usually a process or a add button).

Example of a button adding an extent:

```
<action type="add"
  name="extent"
  or="extent"
  in="/gmd:MD_Metadata/gmd:identificationInfo/gmd:MD_DataIdentification">
  <template>
    <snippet>
      <gmd:extent>
        <gmd:EX_Extent>
          <gmd:geographicElement>
            <gmd:EX_GeographicBoundingBox>
              <gmd:westBoundLongitude>
                <gco:Decimal/>
              </gmd:westBoundLongitude>
              <gmd:eastBoundLongitude>
                <gco:Decimal/>
              </gmd:eastBoundLongitude>
              <gmd:southBoundLatitude>
                <gco:Decimal/>
              </gmd:southBoundLatitude>
              <gmd:northBoundLatitude>
                <gco:Decimal/>
              </gmd:northBoundLatitude>
            </gmd:EX_GeographicBoundingBox>
          </gmd:geographicElement>
        </gmd:EX_Extent>
      </gmd:extent>
    </snippet>
  </template>
</action>
```

(continues on next page)

(continued from previous page)

```

</snippet>
</template>
</action>

```

Example of a button displayed only if there is no resource identifier ending with the metadata record identifier (ie. if attribute) and running the process with add-resource-id identifier:

```

<action type="batch"
  process="add-resource-id"
  if="count (gmd:MD_Metadata/gmd:identificationInfo/*/
    gmd:citation/*/gmd:identifier[
      ends-with(gmd:MD_Identifier/gmd:code/gco:CharacterString,
        //gmd:MD_Metadata/gmd:fileIdentifier/
        ↪gco:CharacterString)]) = 0"/>

```

Attributes:

- **name** (Optional)
- **type** (Optional)

The type of control

- **process** (Optional)

The process identifier (eg. add-resource-id).

- **forceLabel** (Optional)

Force the label to be displayed for this action even if the action is not the first element of its kind. Label will always be displayed.

- **if** (Optional)

An XPath expression to evaluate. If true, the control is displayed. eg.

```

count (gmd:MD_Metadata/gmd:identificationInfo/*/gmd:citation/gmd:CI_Citation/
gmd:identifier[ends-with (gmd:MD_Identifier/gmd:code/gco:CharacterString,
//gmd:MD_Metadata/gmd:fileIdentifier/gco:CharacterString)]) = 0

```

will only displayed the action control if the resource identifier does not end with the metadata identifier.

- **or** (Optional)

Local name to match if the element does not exist.

- **or** (Optional)

The local name of the geonet child (ie. non existing element) to match.

```

<field xpath="/gmd:MD_Metadata/gmd:language"
  or="language"
  in="/gmd:MD_Metadata"/>

```

- **in** (Optional)

XPath of the geonet:child element with the or name to look for. Usually points to the parent of last element of the XPath attribute.

- **in** (Optional)

The element to search in for the geonet child.



- **addDirective** (Optional)

The directive to use for the add control for this field.

Child elements:

- **template**, Optional element (see *Adding a template based field*)

## 1.7 Integrate with CKAN

Sometimes you have two very different groups of users that want to consume your data: one group specialized in spatial data and familiar with OGC standards and another group specialized in open data and familiar with linked data. Some organizations fix this issue by having two different portals with their data synchronized: GeoNetwork and CKAN.

### 1.7.1 Harvest to GeoNetwork

PyCSW can be [added to CKAN as an extension](#) to add a CSW endpoint to CKAN. As GeoNetwork can harvest from any CSW endpoint, we can collect data from CKAN to GeoNetwork easily and keep them synchronized.

Notice that sometimes the conversion from CKAN data formats to CSW may not be complete, so there may be loss of data when harvesting from a CKAN endpoint.

### 1.7.2 Harvest from GeoNetwork

There is a [ckan-geonetwork harvester](#) available that you can install on your CKAN to get all the data available in GeoNetwork also available on a CKAN portal. This way you can use all the power of the metadata editors in GeoNetwork and while making your ckan users happy.

Also, CKAN can harvest CSW endpoints, which can be used to harvest the CSW endpoint of GeoNetwork.

### 1.7.3 Making data available as RDF

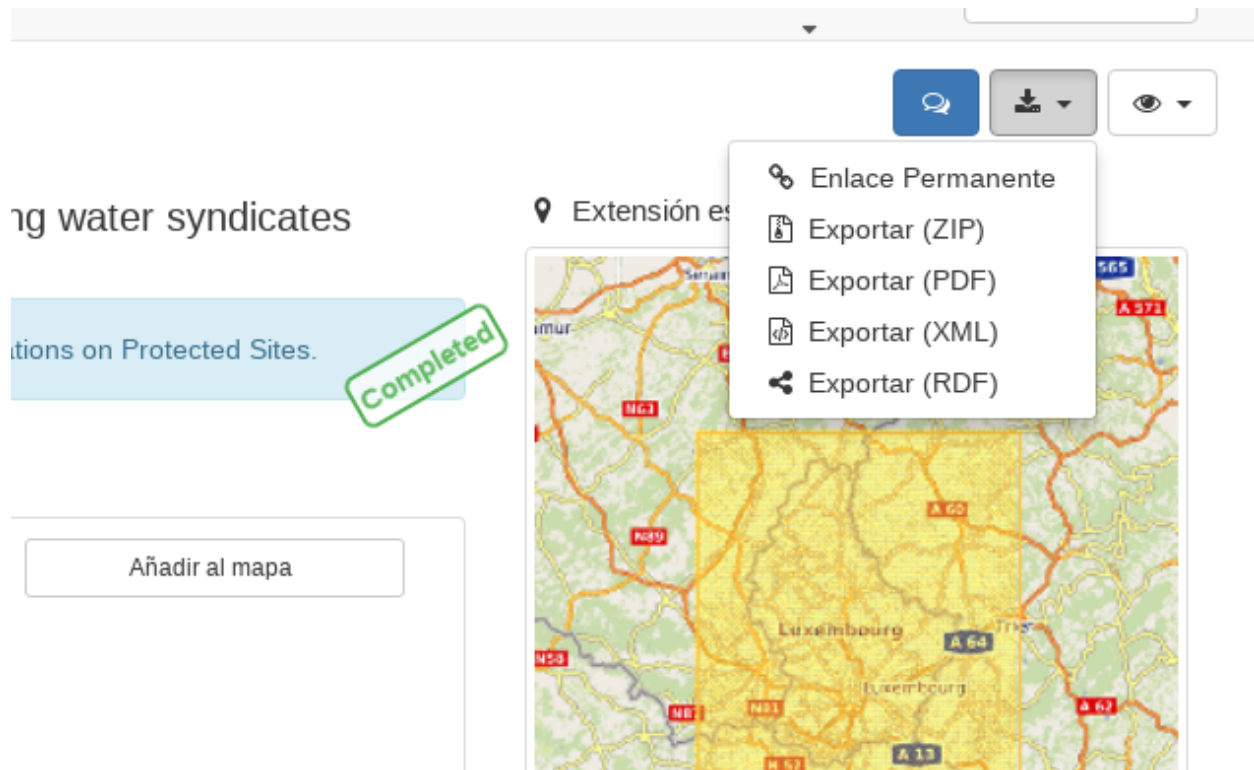
If you like your GeoNetwork user interface, you don't need to sacrifice it in favor of CKAN to have linked data. GeoNetwork allows users to download their data on RDF format.

## 1.8 Migrate from older versions of GeoNetwork

On a frequent basis, we would want to migrate to a newer version of GeoNetwork. Migration between minor versions are easier than migration between major versions, but sometimes you will have to perform a major migration to keep up to date with latest features and bug fixing.

### 1.8.1 Default method

There is a default procedure for migration that simply consist in deploying the new version of GeoNetwork with the old database and data directory. On the next startup, GeoNetwork will detect a new version and should be able to migrate transparently to the user.



**Warning:** Always do a backup of the database and the data directory before attempting any upgrade, as a failed migration can leave the system on an inconsistent state.

But although this method works for most use cases, there is a chance that custom modifications of GeoNetwork may lead to errors on the automatic migration scripts, specially when migrating from 2.x versions.

## 1.8.2 Manual migration

On case of failure of the automatic procedure, we can always try a manual migration. This manual migration can have two different approaches:

### Apply manually the scripts

If the default method fails, a qualified technician can try to apply manually one by one all the migration scripts available on GeoNetwork and check where they fail. It is advisable to migrate step by step only upgrading to the immediately newer version instead of trying to do all the migration in one single step.

**Warning:** Even if the execution of a script does not throw any error, that doesn't mean the migration was successful. Always check that GeoNetwork is working fine and no data was lost in the process.

## Harvest the data

If you are not familiar with the database and the data directory structure of GeoNetwork, it may be a bit complex to manually apply and fix the scripts. On this cases, you can always fallback to harvest all the data from the old GeoNetwork instance to the new instance.

**Warning:** With this method you will lose all the configuration data like harvesters, mail settings and user and groups privileges.

You will not harvest users and groups with this method, so you will have to manually re-create them on the new GeoNetwork instance and make sure metadata is assigned to the proper users and groups.

---

**Note:** It is advisable to ask a qualified technician for a migration script for the custom configuration of the portal.

---