GeoNetwork's Server Reference

Author: **ANDREA CARBONI**

Revision 4 (09-Mar-2007)

Introduction

This document gives some insights about the internal structure of GeoNetwork opensource.

Contents

I Basic Operations **1**

1 Compiling and running GeoNetwork opensource **2**

1.1 System Requirements 2

1.2 Package structure 2

1.3 Source code structure 2

1.4 Compiling GeoNetwork 2

II Protocols **3**

2 Harvesting **4**

III File formats **5**

3 Metadata Exchange Format v1.0 **6**

3.1 Introduction 6

3.2 File format 6

3.3 The info.xml file 7

IV XML Services **10**

4 Calling specifications **11**

5 General services **12**

5.1 xml.info 12

5.2 xml.forward 16

6 Harvesting services **17**

6.1 Introduction 17

6.2 xml.harvesting.get 17

6.3 xml.harvesting.add 19

6.4 xml.harvesting.update 23
6.5 xml.harvesting.remove/start/stop/run 23

7 System configuration **25**
7.1 Introduction 25
7.2 xml.config.get 25
7.3 xml.config.update 27

8 MEF services **28**
8.1 Introduction 28
8.2 mef.export 28
8.3 mef.import 29

V Settings internal structure **30**

9 Settings hierarchy **31**
9.1 Introduction 31
9.2 The system hierarchy 31
9.3 Harvesting nodes 32
9.3.1 Nodes of type geonetwork 32
9.3.2 Nodes of type webFolder 33

Part I

Basic Operations

Chapter 1

Compiling and running GeoNetwork opensource

1.1 System Requirements

1.2 Package structure

1.3 Source code structure

1.4 Compiling GeoNetwork

Part II

Protocols

Chapter 2

Harvesting

Part III

File formats

Chapter 3

Metadata Exchange Format v1.0

3.1 Introduction

The metadata exchange format (**MEF** in short) is a special designed file format whose purpose is to allow metadata exchange between different platforms. A metadata exported into this format can be imported by any platform which is able to understand it. This format has been developed with GeoNetwork in mind so the information it contains is mainly related to it. Nevertheless, it can be used as an interoperability format between any platform.

This format has been designed with these needs in mind:

- Export a metadata record for backup purposes
- Import a metadata record from a previous backup
- Import a metadata record from a different GeoNetwork version to allow a smooth migration from one version to another.

All these operations regard the metadata and its related data as well.

In the paragraphs below, some terms should be intended as follows:

- the term **actor** is used to indicate any system (application, service etc...) that operates on metadata.
- the term **reader** will be used to indicate any actor that can import metadata from a MEF file.
- the term **writer** will be used to indicate any actor that can generate a MEF file.

3.2 File format

A MEF file is simply a ZIP file which contains the following files:

- **metadata.xml** : this file contains the metadata itself, in XML format. The text encoding of the metadata is that one specified into the XML declaration.

- **info.xml** : this is a special XML file which contains information related to the metadata but that cannot be stored into it. Examples of such information are the creation date, the last change date, privileges on the metadata and so on. Now this information is related to the GeoNetwork's architecture.
- **public** : this is a directory used to store the metadata thumbnails and other public files. There are no restrictions on the images' format but it is strongly recommended to use the portable network graphics (PNG), the JPEG or the GIF formats.
- **private** : this is a directory used to store all data (maps, shape files etc...) associated to the metadata. Files in this directory are *private* in the sense that an authorization is required to access them. There are no restrictions on the file types that can be stored into this directory.

Any other file or directory present into the MEF file should be ignored by readers that don't recognize them. This allows actors to add custom extensions to the MEF file.

A MEF file can have empty **public** and **private** folders depending on the export format, which can be:

- **simple** : both public and private are omitted.
- **partial** : only public files are provided.
- **full** : both public and private files are provided.

It is recommended to use the **.mef** extension when naming MEF files.

3.3 The info.xml file

This file contains general information about a metadata. It must have an **info** root element with a mandatory **version** attribute. This attribute must be in the X.Y form, where X represents the major version and Y the minor one. The purpose of this attribute is to allow future changes of this format maintaining compatibility with older readers. The policy behind the version is this:

- A change to Y means a minor change. All existing elements in the previous version must be left unchanged: only new elements or attributes may be added. A reader capable of reading version X.Y is also capable of reading version X.Y' with Y'>Y.
- A change to X means a major change. Usually, a reader of version X.Y is not able to read version X'.Y with X'>X.

The root element must have the following children:

- **general** : a container for general information. It must have the following children:
  - **uuid** : this is the universally unique identifier assigned to the metadata and must be a valid uuid. This element is optional and, when omitted, the reader should generate one. A metadata without a uuid can be imported several times into the same system without breaking uniqueness constraints.
  - **createDate** : This date indicates when the metadata was created.

  - **changeDate** : This date keeps track of the most recent change to the metadata.
  - **siteId** : this is an uuid that identifies the actor that created the metadata and must be a valid uuid.
  - **schema** : Indicates the metadata's schema. The value can be assigned as will but if the schema is one of those describe below, that value must be used:
    * **dublin-core** : A metadata in the dublin core format as described in http://dublincore.org
    * **fgdc-std** : A metadata in the Federal Geographic Data Committee.
    * **iso19115** : A metadata in the ISO 19115 format
    * **iso19139** : A metadata in the ISO 19115/2003 format for which the ISO19139 is the XML encoding.
  - **format** : Indicates the MEF export format. The element's value must belong to the following set: { *simple, partial*, *full* }.
  - **localId** : This is an optional element. If present, indicates the id used locally by the sourceId actor to store the metadata. Its purpose is just to allow the reuse of the same local id when reimporting a metadata.
  - **isTemplate** : A boolean field that indicates if this metadata is a template used to create new ones. There is no real distinction between a real metadata and a template but some actors use it to allow fast metadata creation. The value must be: { *true*, *false* }.

- **categories** : a container for categories associated to this metadata. A category is just a name, like 'audio-video' that classifies the metadata to allow an easy search. Each category is specified by a **category** element which must have a **name** attribute. This attribute is used to store the category's name. If there are no categories, the **categories** element will be empty.

- **privileges** : a container for privileges associated to this metadata. Privileges are operations that a group (which represents a set of users) can do on a metadata and are specified by a set of **group** elements. Each one of these, has a mandatory **name** attribute to store the group's name and a set of **operation** elements used to store the operations allowed on the metadata. Each **operation** element must have a **name** attribute which value must belong to the following set: { *view*, *download*, *edit*, *notify*, *admin*, *dynamic*, *featured* }. If there are no groups or the actor does not have the concept of group, the **privileges** element will be empty. A **group** element without any **operation** element must be ignored by readers.

- **public** : All metadata thumbnails (and any other public file) must be listed here. This container contains a **file** element for each file. Mandatory attributes of this element are **name**, which represents the file's name and **changeDate**, which contains the date of the latest change to the file. The **public** element is optional but, if present, must contain all the files present in the metadata's **public** directory and any reader that imports these files must set the latest change date on these using the provided ones. The purpose of this element is to provide more information in the case the MEF format is used for metadata harvesting.

- **private** : This element has the same purpose and structure of the **public** element but is related to maps and all other private files.

Any other element or attribute should be ignored by readers that don't understand them. This allows actors to add custom attributes or subtrees to the XML.

Figure 3.1 shows an example of info file.

```
<info version="1.0">
   <general>
      <uuid>0619abc0-708b-eeda-8202-000d98959033</uuid>
      <createDate>2006-12-11T10:33:21</createDate>
      <changeDate>2006-12-14T08:44:43</changeDate>
      <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
      <schema>iso19139</schema>
      <format>full</format>
      <localId>204</localId>
      <isTemplate>false</isTemplate>
   </general>
   <categories>
      <category name="maps"/>
      <category name="datasets"/>
   </categories>
   <privileges>
      <group name="editors">
         <operation name="view"/>
         <operation name="download"/>
      </group>
   </privileges>
   <public>
      <file name="small.png" changeDate="2006-10-07T13:44:32"/>
      <file name="large.png" changeDate="2006-11-11T09:33:21"/>
   </public>
   <private>
      <file name="map.zip" changeDate="2006-11-12T13:23:01"/>
   </private>
</info>
```

Figure 3.1: Example of info file

Date format

Unless differently specified, all dates in this file must be in the ISO/8601 format. The pattern must be YYYY-MM-DDTHH:mm:SS and the timezone should be the local one.

Part IV

XML Services

Chapter 4

Calling specifications

tramite http get/post

exceptions

Chapter 5

General services

5.1 xml.info

The xml.info service can be used to query the site about its configuration, services, status and so on. For example, it is used by the harvesting web interface to retrieve information about a remote node.

Request

The xml request should contain at least one **type** element to indicates the kind of information to retrieve. More **type** elements can be specified to obtain more information at once. The set of allowed values are:

site Returns general information about the site like its name, id, etc...

categories Returns all site's categories

groups Returns all site's groups

operations Returns all possible operations on metadata

regions Returns all geographical regions usable for queries

knownNodes Returns all geonetwork nodes that the remote site knows. Every time the remote site harvests from other nodes, it collects information about their address, name, siteId etc... This allows the propagation of the GeoNetwork sites to facilitate harvesting of new sites. Anyway, the remote node may know other nodes by other ways.

harvestingNodes Returns all active nodes the remote node is currently harvesting from. When harvesting from a remote node, this information is usefull to find out how many metadata have been harvested and to which siteId they belong. If there are no metadata from a node, that node will not be returned.

Request example:

```
<request>
   <type>site</type>
   <type>groups</type>
</request>
```

Response

Each **type** element produces an XML subtree so the response to the previous request is like this:

```
<info>
   <site>...</site>
   <categories>...</categories>
   <groups>...</groups>
   ...
</info>
```

Here follows the structure of each subtree:

- **site** : This is the container
  - **name** : Human readable site name
  - **siteId** : Universal unique identifier of the site
  - **platform** : This is just a container to hold the site's backend
    - **name** : Plaform name. For GeoNetwork installations it must be **geonetwork**.
    - **version** : Platform version, given in the X.Y.Z format
    - **subVersion** : Additional version notes, like 'alpha-1' or 'beta-2'.

  Example:

```
<site>
   <name>My site</name>
   <organization>FAO</organization>
   <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
   <platform>
      <name>geonetwork</name>
      <version>2.1.0</version>
   </platform>
</site>
```

- **categories** : This is the container for categories.
  - **category** [0..n] : A single GeoNetwork's category. This element has an **id** attribute which represents the local identifier for the category. It can be usefull to a client to link back to this category.
    - **name** : Category's name
    - **label** : The localized labels used to show the category on screen. See 5.1 below.

  Example:

```
<categories>
   <category id="1">
      <name>datasets</name>
      <label>
         <en>Datasets</en>
         <fr>I don't know</fr>
      </label>
   </category>
</categories>
```

- **groups** : This is the container for groups

  - **group** [2..n] : This is a Geonetwork group. There are at least the internet and intranet groups. This element has an **id** attribute which represents the local identifier for the group.
    * **name** : Group's name
    * **description** : Group's description
    * **referrer** : The user responsible for this group
    * **email** : The email address to notify when a map is downloaded
    * **label** : The localized labels used to show the group on screen. See 5.1 below.

  Example:

```
<groups>
   <group id="1">
      <name>editors</name>
      <label>
         <en>Editors</en>
         <fr>I don't know</fr>
      </label>
   </group>
</groups>
```

- **operations** : This is the container for the operations
  - **operation** [0..n] : This is a possible operation on metadata. This element has an **id** attribute which represents the local identifier for the operation.
    * **name** : Short name for the operation.
    * **reserved** : Can be **y** or **n** and is used to distinguish between system reserved and user defined operations.
    * **label** : The localized labels used to show the operation on screen. See 5.1 below.

  Example:

```
<operations>
   <operation id="0">
      <name>view</name>
      <label>
         <en>View</en>
         <fr>I don't know</fr>
      </label>
   </operation>
</operations>
```

- **regions** : This is the container for geographical regions
  - **region** [0..n] : This is a region present into the system. This element has an **id** attribute which represents the local identifier for the operation.
    * **north** : North coordinate of the bounding box.
    * **south** : South coordinate of the bounding box.
    * **west** : West coordinate of the bounding box.
    * **east** : east coordinate of the bounding box.
    * **label** : The localized labels used to show the region on screen. See 5.1 below.

Example:

```
<regions>
   <region id="303">
      <north>82.99</north>
      <south>26.92</south>
      <west>-37.32</west>
      <east>39.24</east>
      <label>
         <en>Western Europe</en>
         <fr>Western Europe</fr>
      </label>
   </region>
</regions>
```

- **knownNodes** : This is the container
  - **node** [0..n] : A known GeoNetwork node
    - **name** : Node's name
    - **siteId** : Node's unique identifier
    - **host** : Node's host
    - **port** : Node's port. Can be empty
    - **servlet** : Node's servlet. Together with host and port can be used to call services on the remote node.

Example:

```
<knownNodes>
   <node>
      <name>My Host</name>
      <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
      <host>www.myhost.org</host>
      <port/>
      <servlet>myservlet</servlet>
   </node>
</knownNodes>
```

- **harvestingNodes** : This is the container.
  - **node** [0..n] : A remote node GeoNetwork is harvesting from
    - **name** : Node's name
    - **siteId** : Node's unique identifier
    - **metadata** : This is just the number of metadata that have been harvested from the remote node. This is used by the harvesting web interface to provide some hints to the user.

Example:

```
<harvestingNodes>
   <node>
      <name>My Host</name>
      <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
      <metadata>23</metadata>
   </node>
</harvestingNodes>
```

Localized entities

Localized entities have a general **label** element which contains the localized strings in all supported languages. This element has as many children as the supported languages. Each child has a name that reflect the language code while its content is the localized text. Here is an example of such elements:

```
<label>
   <en>Editors</en>
   <fr>I don't know</fr>
   <es>Neither<es>
</label>
```

5.2 xml.forward

This is just a router service. It is used by Javascript code to connect to a remote host because a Javascript program cannot access a machine other than its server. For example, it is used by the harvesting web interface to query a remote host and retrieve the list of site ids.

Request

The service's request has this form:

```
<request>
   <url>...</url>
    <params>...</params>
</knownNodes>
```

where:

url Indicates the remote url to connect to. Usually points to a GeoNetwork's xml service

params This is just a container for the request that must be executed remotely.

Example:

```
<request>
   <url>http://mynode.org:8080/geonetwork/srv/en/xml.info</url>
    <params>
      <request>
         <type>site<type>
      </request>
   </params>
</request>
```

Please notice that this service uses the GeoNetwork's proxy configuration.

Response

The response is just the response of the remote service.

Chapter 6

Harvesting services

6.1 Introduction

This chapter provides a detailed explanation of the GeoNetwork's harvesting services. These services allow a complete control over the harvesting behaviour. They are used by the web interface and can be used by any other client.

6.2 xml.harvesting.get

Retrieves information about one or all configured harvesting nodes.

Request

Called with no parameters returns all nodes. Example:

```
<request/>
```

Otherwise, an **id** parameter can be specified:

```
<request>
   <id>123</id>
</request>
```

Response

When called with no parameters the service provide its output inside a **nodes** container. You get as many **node** elements as are configured. Figure 6.1 shows an example of output.

If you specify an id, you get a response like that one in figure 6.2 (for a web folder node).

The node's structure is described in the next two services. What are missing are the following elements:

- **info** : A container for general information.
  - **lastRun** (*string*) : The lastRun element will be filled as soon as the harvester starts harvesting from this entry. The value is the

```
<nodes>
   <node id="125" name="test 1" type="geonetwork">
      <site>
         <host>localhost</host>
         <port>8080</port>
         <servlet>geonetwork</servlet>
         <account>
            <use>true</use>
            <username />
            <password />
         </account>
      </site>
      <searches>
         <search>
            <freeText />
            <title />
            <abstract />
            <keywords />
            <digital>false</digital>
            <hardcopy>false</hardcopy>
            <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
            <siteName>Food and Agriculture Organization</siteName>
         </search>
      </searches>
      <options>
         <every>90</every>
         <createGroups>true</createGroups>
         <createCateg>true</createCateg>
         <oneRunOnly>false</oneRunOnly>
         <status>inactive</status>
      </options>
      <info>
         <lastRun />
      </info>
   </node>
</nodes>
```

Figure 6.1: Example of an xml.harvesting.get response for a geonetwork node

  - **running** (boolean) : True if the harvester is currently running.

- **error** : This element will be present if the harvester encounters an error during harvesting.
  - **code** : The error code, in string form
  - **message** : The description of the error.
  - **object** : The object that caused the error (if any). This element can be present or not depending on the case.

When getting information about a harvesting node of type Geonetwork, the **search** element may contain a **siteName** child. This element is present and filled when Geonetwork is able to retrieve it from the remote site.

Errors

- **ObjectNotFoundEx** If the **id** parameter is provided but the node cannot be found.

```
<node id="165" name="test" type="webFolder">
   <site>
      <url>http://www.mynode.org/metadata</url>
      <account>
         <use>true</use>
         <username>admin</username>
         <password>admin</password>
      </account>
   </site>
   <options>
      <every>90</every>
      <oneRunOnly>false</oneRunOnly>
      <structure>false</structure>
      <validate>true</validate>
      <status>inactive</status>
   </options>
   <privileges>
      <group id="0">
         <operation name="view" />
      </group>
      <group id="14">
         <operation name="download" />
      </group>
   </privileges>
   <info>
      <lastRun />
   </info>
</node>
```

Figure 6.2: Example of an xml.harvesting.get response for a web folder node

6.3 xml.harvesting.add

Create a new harvesting entry. The entry can be of any type support by GeoNetwork (GeoNetwork node, web folder etc...). When a new entry is created, its status is set to **inactive**. A call to the xml.harvesting.start service is required to start harvesting.

Request

The service requires an XML tree with all information the client wants to add.

To create a GeoNetwork entry the following XML information should be provided. Default values are given in parenthesis (after the parameter's type) and are used when the parameter is omitted. If the type is boolean, only the **true** and **false** strings are allowed.

- **node** : This is the root container. The **name** attribute specifies the entry's name. If omitted, an empty string is considered. The **type** attribute is mandatory and must be **geonetwork**.
  - **site** : A container for site information. Can be omitted.
    - **host** (*string*, '') : The GeoNetwork node's host name or IP address.
    - **port** (*string*, '80') : The port to connect to.
    - **servlet** (*string*, 'geonetwork'). The servlet name choosen in the remote site.
  - **account** : A container for account information. Can be omitted.

    - **use** (*boolean*, 'false') : True means that the harvester will use the provided username and password to authenticate itself.
    - **username** (*string*, '') : Username on the remote node.
    - **password** (*string*, '') : Password on the remote node.
  - **searches** : A container for search parameters. Can be omitted.
    - **search** : A container for a single search on a siteID. You can specify 0 or more searches but to properly harvest you have to provide at least one.
      - **freeText** (*string*, '') : Free text to search. This and the following parameters are the same used during normal search using the web interface.
      - **title** (*string*, '')
      - **abstract** (*string*, '')
      - **keywords** (*string*, '')
      - **digital** (*boolean*, 'false')
      - **hardcopy** (*boolean*, 'false')
      - **siteId** (*string*,) : One of the siteIDs present on the remote node. Must be present and not empty.
  - **options** : A container for generic options. Can be omitted.
    - **every** (*integer*, '90') : Harvesting interval in minutes.
    - **createGroups** (*boolean*, 'true') : If true, GeoNetwork tries to keep remote privileges creating local groups. After the harvesting, the local node will have all the remote node's groups. A group is created locally only if it does not already exist. The group's name is used to establish that.
    - **createCateg** (*boolean*, 'true') : If true, GeoNetwork will create local categories to match the remote ones. Normally, if a remote category does not exist locally, the link between that category and the metadata is lost.
  - **oneRunOnly** (*boolean*, 'false') : After the first run, the entry's status will be set to **inactive**.

Please note that even if clients can store empty values ('') for many parameters, before starting the harvesting entry those parameters should be properly set in order to avoid errors.

Figure 6.3 shows an example of an XML request to create a GeoNetwork entry.

To create a WebFolder entry, the following XML information should be provided:

- **node** : This is the root container. The **name** attribute specifies the entry's name. If omitted, an empty string is considered. The **type** attribute is mandatory and must be **webFolder**.
  - **site** : A container for site information. Can be omitted.
    - **url** (*string*, '') : The URL to harvest from. If provided, must be a valid URL starting with 'HTTP://'.
    - **account** : A container for account information. Can be omitted.
      - **use** (*boolean*, 'false') : If true, GeoNetwork will use the basic HTTP authentication.
      - **username** (*string*, '') : Username on the remote node.
      - **password** (*string*, '') : Password on the remote node.

```
<node name="South Africa" type="geonetwork">
   <site>
      <host>south.africa.org</host>
      <port>8080</port>
      <servlet>geonetwork</servlet>
      <account>
         <use>true</use>
         <username>admin</username>
         <password>admin</password>
      </account>
   </site>
   <searches>
      <search>
         <freeText />
         <title />
         <abstract />
         <keywords />
         <digital>true</digital>
         <hardcopy>false</hardcopy>
         <siteId>0619cc50-708b-11da-8202-000d9335906e</siteId>
      </search>
   </searches>
   <options>
      <every>90</every>
      <createGroups>true</createGroups>
      <createCateg>true</createCateg>
      <oneRunOnly>false</oneRunOnly>
   </options>
</node>
```

Figure 6.3: Example of an xml.harvesting.add request for a geonetwork node

```
<node name="Asia remote node" type="webFolder">
   <site>
      <url>http://www.mynode.org/metadata</url>
      <account>
         <use>true</use>
         <username>admin</username>
         <password>admin</password>
      </account>
   </site>
   <options>
      <every>90</every>
      <oneRunOnly>false</oneRunOnly>
      <structure>false</structure>
      <validate>true</validate>
      <status>inactive</status>
   </options>
   <privileges>
      <group id="0">
         <operation name="view" />
      </group>
      <group id="14">
         <operation name="download" />
      </group>
   </privileges>
</node>
```

Figure 6.4: Example of an xml.harvesting.add request for a web folder node

- **privileges** : A container for privileges that must be associated to the harvested metadata. Can be omitted but doing so the harvested metadata will not be visible.
  - **group** : A container for the privileges to associate to this group. It must have an **id** attribute which value should be the identifier of a GeoNetwork group. If the id is not a valid group id, the contained privileges will be discarded.
    - **operation** : Specifies an operation to associate to the containing group. It must have a **name** attribute which value can be one of : **view**, **download**, **dynamic**, **notify**, **featured**.
- **options** : A container for generic options. Can be omitted.
  - **every** (*integer*, '90') : Harvesting interval in minutes.
  - **oneRunOnly** (*boolean*, 'false') : After the first run, the entry's status will be set to **inactive**.
  - **structure** (*boolean*, 'false') : When true, the folders present at the remote host will provide additional information about categories and siteIDs.
  - **validate** (*boolean*, 'false') : When true, GeoNetwork will validate every metadata against its schema. If the metadata is not valid, it will not be imported.

Figure 6.4 shows an example of an XML request to create a web folder entry.

Response

The service's response is the output of the **xml.harvesting.get** service of the newly created entry. So, the response is similar to the request but with some more information:

- An **id** attribute to the **node** root element. This is the internal identifier that must be used to update/start/stop the entry.
- An **info** element with some information and statistics.

6.4 xml.harvesting.update

This service is responsible for changing the entry's parameters. A typical request has a **node** root element and must include the **id** attribute:

```
<node id="24">
   ...
</node>
```

The body of the **node** element depends on the node's type. The update policy is this:

- If an element is specified, the associated parameter is updated.
- If an element is not specified, the associated parameter will not be changed.

So, you need to specify only the elements you want to change. There are some exceptions for the following node types:

geonetwork The only exception to this behaviour is the **searches** element: if omitted searches will not be changed but if specified all contained search entries will replace the old ones.

webFolder The **privileges** element works as the previous one: if omitted privileges will not be changed but if specified all current privileges will be replaced by the provided ones.

Note that you cannot change the type of an entry once it has been created.

Request

The request is the same as that used to add an entry. Only the **id** attribute is mandatory.

Response

The response is the same as the **xml.harvesting.get** called on the updated entry.

6.5 xml.harvesting.remove/start/stop/run

These services are put together because they share a common request interface. Their purpose is obviously to remove, start, stop or run a harvesting entry.

Request

A set of **id**s to operate on. Example:

```
<request>
   <id>123</id>
   <id>456</id>
   <id>789</id>
</request>
```

If the request is empty, nothing is done.

Response

The same as the request but every id has a status attribute indicating the success or failure of the operation. For example, the response to the previous request could be:

```
<request>
   <id status="ok">123</id>
   <id status="not-found">456</id>
   <id status="inactive">789</id>
</request>
```

The following table summarizes, for each service, the possible status values:

Status value	remove	start	stop	run
ok	+	+	+	+
not-found	+	+	+	+
inactive	-	-	-	+
already-inactive	-	-	+	-
already-active	-	+	-	-
already-running	-	-	-	+

Chapter 7

System configuration

7.1 Introduction

The GeoNetwork's configuration is made up of a set of parameters that can be changed to accomodate any installation need. These parameters are subdivided into 2 groups:

- parameters that can be easily changed through a web interface.
- parameters not accessible from a web interface and that must be changed when the system is not running.

The first group of parameters can be queried or changed through 2 services: **xml.config.get** and **xml.config.update**.

7.2 xml.config.get

This service returns the system configuration's parameters.

Request

No parameters are needed.

Response

The response is an XML tree similar to the **system** hyerarchy into the settings structure. See 9.2 for more information. The response has the following elements:

- site
  - name
  - organization
- server

  - host
  - port
- intranet
  - network
  - netmask
- z3950
  - enable
  - port
  - proxy
- use
  - host
  - port
- feedback
  - email
  - mailServer
    - host
    - port

```
<config>
   <site>
      <name>dummy</name>
      <organization>dummy</organization>
   </site>
   <server>
      <host>localhost</host>
      <port>8080</port>
   </server>
   <intranet>
      <network>127.0.0.1</network>
      <netmask>255.255.255.0</netmask>
   </intranet>
   <z3950>
      <enable>true</enable>
      <port>2100</port>
   </z3950>
   <proxy>
      <use>false</use>
      <host/>
      <port/>
   </proxy>
   <feedback>
      <email/>
      <mailServer>
         <host/>
         <port>25</port>
      </mailServer>
   </feedback>
</config>
```

7.3 xml.config.update

Request

Response

Chapter 8

MEF services

8.1 Introduction

This chapter describes the services related to the Metadata Exchange Format (see chapter 3). These services allow to import/export metadata using the MEF format.

8.2 mef.export

As the name suggests, this service exports a GeoNetwork's metadata using the MEF file format.

This service is public but metadata access rules apply. For a partial export, the **view** privilege is enough but for a full export the **download** privilege is also required. Without a login step, only partial exports on public metadata are allowed.

This service uses the system's temporary directory to build the MEF file. With full exports of big data maybe it is necessary to change this directory. In this case, use the Java's -D command line option to set the new directory before running GeoNetwork (if you use Jetty, simply change the script into the **bin** directory).

Request

This service accepts requests in GET/POST and XML form. The input parameters are:

uuid the universal unique identifier of the metadata

format which format to use. Can be one of : **simple**, **partial**, **full**.

skipUuid If provided, tells the exporter to not export the metadata's uuid. Without the uuid (which is a unique key inside the database) the metadata can be imported over and over again. Can be one of: **true**, **false**.

Response

The service's response is a MEF file with these characteristics:

- the name of the file is the metadata's uuid
- the extension of the file is **mef**

8.3 mef.import

This service is reserved to administrators and is used to import a metadata provided in the MEF format.

Request

The service accepts a multipart/form-data POST request with a single **mefFile** parameter that must contain the MEF information.

Response

If all goes well, the service returns an **ok** element containing the local id of the created metadata. Example:

```
<ok>123</ok>
```

Part V

Settings internal structure

Chapter 9

Settings hierarchy

9.1 Introduction

GeoNetwork stores many options and information inside the **Settings** table. Information is grouped into hierarchies where each node has a key/value pair and can have many children. Each key is limited to 32 characters while each value is limited to 250. The 2 top level hierarchies are **system** and **harvesting**.

In the following sections, the indentation is used to show hierarchies. Names in bold represent keys with the value's datatype in parenthesys. An *italic* font is used to indicate basic types (string, integer, boolean) while normal font with a | is used to represent a set of allowed values. Regarding the boolean type, value can be only **true** or **false**. A missing datatype means that the value of the node is not used. Square brackets indicate cardinality. If they are missing, a cardinality of [1..1] should be considered.

9.2 The system hierarchy

- **site** : Contains information about the site
  - **name** (*string*) : Name used to present this site to other sites. Used to fill comboboxes or lists.
  - **organization** (*string*) : Name of the organization/company/institute that is running GeoNetwork
  - **siteId** (*string*) : A UUID that uniquely identifies the site. It is generated by the installer.
- **platform** : Contains information about the current version
  - **version** (*string*) : GeoNetwork's version in the X.Y.Z format
  - **subVersion** (*string*) : A small description about the version, like 'alpha-1', 'beta' etc...
- **server** : Used when it is necessary to build absolute URLs to the GeoNetwork server. This is the case, for example, when creating links inside a metadata or when providing CS/W capabilities.
  - **host** (*string*) : Main HTTP server's address
  - **port** (*integer*) : Main HTTP server's port (can be empty)

- **intranet** : specify the network of the intranet
  - **network** (*string*) : Network's address
  - **netmask** (*string*) : Network's netmask
- **z3950** : A container for Z39.50 server parameters
  - **enable** (*boolean*) : If true, GeoNetwork will start the Z30.50 server
  - **port** (*integer*) : The port opened by GeoNetwork to listen to Z39.50 requests. Usually is 2100.
- **proxy** : This container speficy proxy configuration to use
  - **use** (*boolean*) : If true, GeoNetwork will use the given proxy for outgoing connections
  - **host** (*string*) : Proxy host
  - **port** (*integer*) : Proxy port
- **feedback** : Feedback is sent with proper web form or when downloading a resource.
  - **email** (*string*) : email address of a GeoNetwork administrator or someone else
  - **mailServer** : This container represents the mail server that will be used to send emails
    - **host** (*string*) : Address of the SMTP server to use
    - **port** (*string*) : SMTP port to use

9.3 Harvesting nodes

The second top level hierarchy is **harvesting**. All nodes added using the web interface are stored here. Each child has **node** in its key and its value can be **geonetwork** or **webFolder** depending on the node's type.

9.3.1 Nodes of type geonetwork

- **site** : Contains host and account information
  - **name** (*string*) : Node's name to show in the list
  - **host** (*string*)
  - **port** (*integer*)
  - **servlet** (*string*)
  - **useAccount** (*boolean*)
    - **username** (*string*)
    - **password** (*string*)
- **search** [0..n] : Contains the search parameters. If this element is missing no harvesting will be performed but the host's parameters will be used to connect to the remote node.

  - **freeText** (*string*)
  - **title** (*string*)
  - **abstract** (*string*)
  - **keywords** (*string*)
  - **digital** (*boolean*)
  - **hardcopy** (*boolean*)
  - **siteId** (*string*)

- **options**

  - **every** (*integer*) : Timeout, in minutes, between 2 consecutive harvesting.
  - **createGroups** (*boolean*) : If true, the harvester will create locally the same groups of the harvested metadata in order to keep remote privileges. Remote groups that already exist locally are not created.
  - **createCateg** (*boolean*) : If true, the harvester will create locally the same categories of the harvested metadata in order to keep the link to the metadata. Remote categories that already exist locally are not created.
  - **oneRunOnly** (*boolean*) : If true, the harvester will harvest one time from this node and then it will set the status to inactive.
  - **status** (activelinactive) : Indicates if the harvesting from this node is stopped (inactive) or if the harvester is waiting until the timeout comes.

- **info** : Just a container for some information about harvesting from this node

  - **lastRun** (*string*) : If not empty, tells when the harvester harvested from this node

9.3.2 Nodes of type webFolder

- **site** : Contains the URL to connect to and account information

  - **name** (*string*) : Node's name to show in the list
  - **url** (*string*) : URL to connect to. Must be well formed, starting with 'http://', 'file://' or a supported protocol
  - **useAccount** (*boolean*) : Indicates if the harvester has to authenticate to access the data. Only the http basic authentication is used.
    - **username** (*string*)
    - **password** (*string*)

- **options**

  - **every** (*integer*) : Timeout, in minutes, between 2 consecutive harvesting.
  - **oneRunOnly** (*boolean*) : If true, the harvester will harvest one time from this node and then it will set the status to inactive.
  - **structure** (*boolean*) : Indicates if remote folder is flat or it contains subfolder with a precise meaning. See the administrator's guide.

  - **validate** (*boolean*) : If set, the harvester will validate the metadata against its schema and the metadata will be harvested only if it is valid.
  - **status** (activelinactive) : Indicates if the harvesting from this node is stopped (inactive) or if the harvester is waiting until the timeout comes.

- **privileges** : This is a container for privileges to assign to each imported metadata
  - **group** (*integer*) [0..n] : Indicate a local group. The node's value is its local identifier. There can be several group nodes each with its set of privileges
    - **operation** (*integer*) [0..n] : Privilege to assign to the group. The node's value is the numeric id of the operation like 0=view, 1=download, 2=edit etc...

- **info** : Just a container for some information about harvesting from this node
  - **lastRun** (*string*) : If not empty, tells when the harvester harvested from this node. The value is the current time in millis since 1 January, 1970.