



deegree Web Catalogue Service v.2.5

lat/lon GmbH

Aennchenstr. 19
53177 Bonn
Germany
Tel ++49 - 228 - 184 96-0
Fax ++49 - 228 - 184 96-29
info@lat-lon.de
www.lat-lon.de

Dept. of Geography
Bonn University
Meckenheimer Allee 166
53115 Bonn

Tel. ++49 228 732098

Change log

Date	Description	Author
2008-06-10	Created new documentation for deegree CSW 2.2 supporting OpenGIS Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile (1.0.0) based on documentation of deegree CSW 2.1.	Lyn Büsching
2008-12-08	Added configuration hints to detect invalid service metadatasets.	Lyn Büsching
2009-07-28	Added configuration hint (URL of the CSW in the file csw202.properties)	Lyn Büsching
2009-09-03	Added Harvester Functionality	Lyn Büsching
2009-12-02	Update to v2.3: mainly replaced Appendix A and B with current versions; updated file paths	Sebastian Goerke
2010-01-20	Add notice for harvesting functionality	Sebastian Goerke

Table of Contents

1 Introduction.....	5
2 Download / Installation.....	7
2.1 Prerequisites.....	7
2.2 deegree Web Catalogue Service release.....	7
2.3 Setting up the example database.....	7
2.4 Testing the installation.....	9
3 Architecture.....	16
4 Basic configuration.....	17
4.1 Structure of the configuration files.....	17
4.2 The deegree CSW configuration document.....	17
4.2.1 deegree-Parameters.....	17
4.2.2 ServiceIdentification	18
4.2.3 ServiceProvider.....	19
4.2.4 OperationsMetadata.....	19
4.2.5 Filter_Capabilities.....	20
5 Advanced configuration.....	21
5.1 Manual Tomcat integration.....	21
5.2 Setting up the Harvester.....	24
5.2.1 Setting up the database.....	24
5.2.2 Activating harvesting functionality.....	25
5.2.3 Testing the harvester.....	26
5.3 Request and response formatting using XSLT.....	29
5.3.1 Input/request	30
5.3.2 output/response.....	36
Appendix A Example CSW configuration document.....	40
Appendix B deegree WFS configuration for accessing example metadatasets.....	44

Index of Tables

Table 1: Directory structure of the CSW release.....	7
--	---

Illustration Index

Figure 1: deegree CSW architecture.....	16
Figure 2 File dependencies for deegree CSW configuration.....	17
Figure 3: Tomcat console on a Windows OS.....	26

1 Introduction

deegree is a Java Framework offering the main building blocks for Spatial Data Infrastructures (SDIs). Its entire architecture is developed using standards of the Open Geospatial Consortium (OGC) and ISO Technical Committee 211 – Geographic information / Geoinformatics (ISO/TC 211). deegree encompasses OGC Web Services as well as clients. deegree is Free Software protected by the GNU Lesser General Public License (GNU LGPL) and is accessible at <http://www.deegree.org>.

deegree2 is the new release of deegree supporting a number of features that deegree1 was not able to handle. This documentation describes setup and configuration of the deegree Web Catalogue Service (CSW), an implementation of OGC's Catalogue Service Implementation Specification 2.0.2 and the ISO 19115 / 19119 Application Profile 1.0.0.

deegree's Web Catalogue Service implementation (Catalogue Service – Web profile, therefore CSW) is able to serve different metadata formats in parallel based on the same physical datastore. This is possible because deegree CSW uses XSLT processing to transform requests as well as responses into the desired format. deegree CSW does not contain a data access modul of its own. It uses an OGC WFS (at the moment limited to deegree WFS) as datasource. So in future it will be possible to use deegree CSW on top of any other OGC compliant WFS to offer catalogue functionalities.

As deegree WFS is extremely flexible in regard to supporting different database schemas, it is possible to adjust deegree CSW to almost any existing metadata base – without the need for replication.

Besides the CSW, deegree comprises a number of additional services and clients. A complete list of deegree components can be found at:

<http://www.lat-lon.de> → Products

Downloads of packaged deegree components can be found at:

<http://www.deegree.org> → Download

deegree's Web Catalogue Service offers great flexibility regarding it's configuration and output formats. It works with a wide range of physical data sources and server environments. The configuration of CSW is similar to the configuration of other deegree Web Services and requires the editing of XML files which control the functionality of the server.

deegree CSW is an implementation of the OpenGIS® Catalogue Services Specification 2.0.2 - ISO 19115/ISO19119 Metadata Application Profile (1.0.0) (in short: ISO APP; OGC project document 07-045).

It supports the operations:

- GetCapabilities
- DescribeRecord
- GetRecords
- GetRecordByID
- Transaction and
- Harvest

It even extends the Harvest-operation by the possibility to harvest other Catalogue Services (version 2.0.2) as a whole (a behaviour not specified by the ISO APP and the CS-W 2.0.2 specification).

The web services of deegree are realized as Java modules controlled by one central servlet (the “dispatcher”). This servlet has to be deployed to the respective web server/servlet engine. Most of the common web servers support servlet technology, thus making deegree a universal product. The Apache-Tomcat 5.5 Servlet-Engine is recommended due to its widespread use and its status as an open-source product.

2 Download / Installation

2.1 Prerequisites

For deegree2 Web Catalogue Service to run you need:

- Java (JRE or JDK) version 1.5.x
- Tomcat 5.5.x
- PostgreSQL 8.0 + PostGIS 1.0.x

For installation of these components refer to the corresponding documentation at java.sun.com tomcat.apache.org, <http://www.postgresql.org/> and <http://www.postgis.org/>.

2.2 deegree Web Catalogue Service release

deegree Web Catalogue Service can be downloaded from <http://www.deegree.org>. The release is packed as a WAR-archive. Simply put this file into your `$TOMCAT_HOME$/webapps` directory and (re-)start Tomcat. The installation of deegree WFS is already done with this.

Note: It is also possible to extract the WAR archive into another place of your computer and direct Tomcat to this place. Because of this possibility, in the remainder of this document, the directory you extracted the files to is referred to as `csw_home` (`= $TOMCAT_HOME$/webapps/deegree-csw` in the standard case).

Your `csw_home` will contain the following structure:

directory	Content
./client	contains the generic client for testing
./WEB-INF	Required by Tomcat, containing all libraries, configuration- and data-files
./WEB-INF/conf/csw	WFS configuration files
./WEB-INF/scripts/sql	contains DB create scripts, Format converter ESRI metadata file -> ISO

Table 1: Directory structure of the CSW release

2.3 Setting up the example database

Create a postgis enabled database ('csw') within PostgreSQL with UTF-8 Encoding (consult the according PostgreSQL and PostGIS documentation). After that execute the according SQL scripts **create_database.sql** and afterwards **create_searchTables.sql** located in `csw_home/WEB-INF/scripts/sql`. To create

the indices also execute **create_normal_index.sql** and **create_spatial_index.sql**.

Then you have to connect the catalogue against the PostgreSQL instance in the file WEB-INF/conf/csw/featuretypes/csw_postgres.xsd like this, where csw is the name of your database:

```
<xsd:appinfo>
  <deegreewfs:Prefix>app</deegreewfs:Prefix>
  <deegreewfs:Backend>POSTGIS</deegreewfs:Backend>
  <deegreewfs:DefaultSRS>EPSG:4326</deegreewfs:DefaultSRS>
  <JDBCConnection xmlns="http://www.deegree.org/jdbc">
    <Driver>org.postgresql.Driver</Driver>
    <Url>jdbc:postgresql://localhost:5432/csw</Url>
    <User>USERNAME</User>
    <Password>PASSWORD</Password>
    <SecurityConstraints/>
    <Encoding>iso-8859-1</Encoding>
  </JDBCConnection>
  <deegreewfs:SuppressXLinkOutput>>true</deegreewfs:SuppressXLinkOutput>
</xsd:appinfo>
```

Furthermore it is required to adjust the file WEB-INF/classes/org/deegree/ogcwebservices/csw/csw202.properties. The database connection must be the same as in the WFS featuretype definition above. The deegree CSW 2.0.2 supports coupling of metadatasets with hierarchyLevel 'dataset' or 'series' with service metadatasets (hierarchyLevel: 'service') by reference. The service metadataset and coupled resource must be available in the same instance of the CSW. The URL of the CSW must be entered in the csw202.properties file, it is required to set the reference of a coupled resource in a service metadataset.

To inform the admin about inconsistent service metadatasets which have deleted coupled resources, it is needed to add an email address and smtp host. Run org/deegree/ogcwebservices/csw/DetectInvalidCouplings from the deegree2.jar to detect invalid service metadatasets and inform the admin. After starting the script it will look every 24 hours for invalid metadata sets.

```
# set the database connection as in the internal used WFS featuretype
definition
db.driver=org.postgresql.Driver
db.url=jdbc:postgresql://localhost:5432/csw
db.user=USERNAME
db.password=PASSWORD

# url of the csw
csw.url=http://localhost:8080/deegree-csw/services

# email address and smtp host to inform about inconsistent service
metadatasets
admin.email=admin@deegree-csw.de
smtp host=smtp.host.de
```


2.4 Testing the installation

The deegree CSW demo download comes pre-configured for accessing a Postgis database. The database can be filled with a few example metadataset to enable to perform a few example requests. Metadata datasets will find it's way into the database via a Transaction Insert request which can be sent via the generic client located under the url **http://yourservername:port/deegree-csw/client/client.html**. Go to the 'Request' dropdown menu, and execute all 'Transaction' -> 'insert*.xml' requests. For further testing you have two options. Some example requests for demonstration of the principle behavior of the service will be explained in the following sections. You can also use the generic client to communicate with the CSW. Just check out the example requests in the dropdown menu.

request 1:

```
http://my.server.domain/deegree-csw/services?
SERVICE=CSW&ACCEPTVERSION=2.0.2&REQUEST=GetCapabilities
```

result:

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Capabilities xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:ows="http://www.opengis.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink"
updateSequence="0" version="2.0.2">
  <ows:ServiceIdentification>
    <ows:Title>deegree demo CSW</ows:Title>
    <ows:Abstract>abstract</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>deegree</ows:Keyword>
    </ows:Keywords>
    <ows:ServiceType>CSW</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
    <ows:Fees>NO FEES - IT'S FREE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>lat/lon GmbH</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://www.latlon.de" xlink:type="simple"/>
    <ows:ServiceContact>
      <ows:IndividualName/>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+49 228 18496-0</ows:Voice>
          <ows:Facsimile>+49 228 18496-29</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>Aennchenstr. 19</ows:DeliveryPoint>
          <ows:DeliveryPoint>basement</ows:DeliveryPoint>
          <ows:City>Bonn</ows:City>
          <ows:AdministrativeArea>NRW</ows:AdministrativeArea>
          <ows:PostalCode>53177</ows:PostalCode>
          <ows:Country>Germany</ows:Country>
          <ows:ElectronicMailAddress>info@lat-lon.de</ows:ElectronicMailAddress>
        </ows:Address>
        <ows:OnlineResource xlink:href="http://localhost:8080/deegree-
csw/services" xlink:type="simple"/>
        <ows:HoursOfService>9am-17pm</ows:HoursOfService>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>
</csw:Capabilities>
```

```

        <ows:ContactInstructions>personal</ows:ContactInstructions>
    </ows:ContactInfo>
    <ows:Role>PointOfContact</ows:Role>
</ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
            </ows:HTTP>
        </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeRecord">
        <ows:DCP>
            <ows:HTTP>
                <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
                <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="typeName">
            <ows:Value>csw:dataset</ows:Value>
            <ows:Value>csw:datasetcollection</ows:Value>
            <ows:Value>csw:application</ows:Value>
            <ows:Value>csw:service</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputFormat">
            <ows:Value>application/xml</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="schemaLanguage">
            <ows:Value>XMLSCHEMA</ows:Value>
        </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetRecords">
        <ows:DCP>
            <ows:HTTP>
                <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="typeName">
            <ows:Value>csw:Record</ows:Value>
            <ows:Value>gmd:MD_Metadata</ows:Value>
            <ows:Value>csw:service</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputFormat">
            <ows:Value>application/xml</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="outputSchema">
            <ows:Value>DublinCore</ows:Value>
            <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
            <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="resultType">
            <ows:Value>RESULTS</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="ElementSetName">
            <ows:Value>brief</ows:Value>
            <ows:Value>summary</ows:Value>
            <ows:Value>full</ows:Value>
        </ows:Parameter>
        <ows:Parameter name="CONSTRAINTLANGUAGE">
            <ows:Value>Filter</ows:Value>
            <ows:Value>CQL</ows:Value>
        </ows:Parameter>
    </ows:Operation>

```

```

<ows:Operation name="GetRecordById">
  <ows:DCP>
    <ows:HTTP>
      <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
      <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
    </ows:HTTP>
  </ows:DCP>
</ows:Operation>
<ows:Operation name="Transaction">
  <ows:DCP>
    <ows:HTTP>
      <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
    </ows:HTTP>
  </ows:DCP>
  <ows:Parameter name="outputSchema">
    <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
  </ows:Parameter>
</ows:Operation>
<ows:Constraint name="IsoProfiles">
  <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
</ows:Constraint>
<ows:Constraint name="PostEncoding">
  <ows:Value>SOAP</ows:Value>
</ows:Constraint>
</ows:OperationsMetadata>
<ogc:Filter_Capabilities>
  <ogc:Spatial_Capabilities>
    <ogc:GeometryOperands>
      <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
    </ogc:GeometryOperands>
    <ogc:SpatialOperators>
      <ogc:SpatialOperator name="BBOX"/>
    </ogc:SpatialOperators>
  </ogc:Spatial_Capabilities>
  <ogc:Scalar_Capabilities>
    <ogc:LogicalOperators/>
    <ogc:ComparisonOperators>
      <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
    </ogc:ComparisonOperators>
    <ogc:ArithmeticOperators>
      <ogc:SimpleArithmetic/>
    </ogc:ArithmeticOperators>
  </ogc:Scalar_Capabilities>
  <ogc:Id_Capabilities>
    <ogc:EID/>
  </ogc:Id_Capabilities>
</ogc:Filter_Capabilities>
</csw:Capabilities>

```

request2:

```

http://my.server.domain/deegree-csw/services?
SERVICE=CSW&ACCEPTVERSION=2.0.2&REQUEST=GetCapabilities&Sections=OperationsMetadata

```

result:

```

<?xml version="1.0" encoding="UTF-8"?>
<csw:Capabilities xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"

```

```

xmlns:ows="http://www.opengis.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink"
updateSequence="0" version="2.0.2">
  <ows:OperationsMetadata>
    <ows:Operation name="GetCapabilities">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
    </ows:Operation>
    <ows:Operation name="DescribeRecord">
      <ows:DCP>
        <ows:HTTP>
          <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
          <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="typeName">
        <ows:Value>csw:dataset</ows:Value>
        <ows:Value>csw:datasetcollection</ows:Value>
        <ows:Value>csw:application</ows:Value>
        <ows:Value>csw:service</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="outputFormat">
        <ows:Value>application/xml</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="schemaLanguage">
        <ows:Value>XMLSCHEMA</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetRecords">
      <ows:DCP>
        <ows:HTTP>
          <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
        </ows:HTTP>
      </ows:DCP>
      <ows:Parameter name="typeName">
        <ows:Value>csw:Record</ows:Value>
        <ows:Value>gmd:MD_Metadata</ows:Value>
        <ows:Value>csw:service</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="outputFormat">
        <ows:Value>application/xml</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="outputSchema">
        <ows:Value>DublinCore</ows:Value>
        <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
        <ows:Value>http://www.opengis.net/cat/csw/2.0.2</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="resultType">
        <ows:Value>RESULTS</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="ElementSetName">
        <ows:Value>brief</ows:Value>
        <ows:Value>summary</ows:Value>
        <ows:Value>full</ows:Value>
      </ows:Parameter>
      <ows:Parameter name="CONSTRAINTLANGUAGE">
        <ows:Value>Filter</ows:Value>
        <ows:Value>CQL</ows:Value>
      </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetRecordById">
      <ows:DCP>
        <ows:HTTP>

```

```

        <ows:Get xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
        <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
        </ows:HTTP>
    </ows:DCP>
</ows:Operation>
<ows:Operation name="Transaction">
    <ows:DCP>
        <ows:HTTP>
            <ows:Post xlink:href="http://localhost:8080/deegree-csw/services"
xlink:type="simple"/>
            </ows:HTTP>
        </ows:DCP>
        <ows:Parameter name="outputSchema">
            <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
        </ows:Parameter>
    </ows:Operation>
<ows:Constraint name="IsoProfiles">
    <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
</ows:Constraint>
<ows:Constraint name="PostEncoding">
    <ows:Value>SOAP</ows:Value>
</ows:Constraint>
</ows:OperationsMetadata>
</csw:Capabilities>

```

request3:

```

http://my.server.domain/deegree-csw/services?
request=DescribeRecord&version=2.0.2&outputFormat=text/xml&schemaLanguage=XMLSCHEMA&typeName=csw:Record&namespace=csw:http://www.opengis.org/csw

```

result:

```

<?xml version="1.0" encoding="UTF-8"?>
<csw:DescribeRecordResponse xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/cat/csw/2.0.2
D:\java\projekte\iso19115\CSW202_APIISO_100_FINAL\csw202_apiso100\csw\2.0.2\CSW-
discovery.xsd">
    <csw:SchemaComponent schemaLanguage="XMLSCHEMA"
targetNamespace="http://purl.org/dc/elements/1.1/">
        <xs:schema xmlns="http://purl.org/dc/elements/1.1/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified"
elementFormDefault="qualified" targetNamespace="http://purl.org/dc/elements/1.1/">
            <xs:annotation>
                <xs:documentation>
                    Simple DC XML Schema, 2002-03-12
                </xs:documentation>
            </xs:annotation>
        </xs:schema>
    </csw:SchemaComponent>

```

```

by Pete Johnston (p.johnston@ukoln.ac.uk),
Carl Lagoze (lagoze@cs.cornell.edu), Andy Powell (a.powell@ukoln.ac.uk),
Herbert Van de Sompel (hvdsomp@yahoo.com).
This schema defines terms for Simple Dublin Core, i.e. the 15
elements from the http://purl.org/dc/elements/1.1/ namespace, with
no use of encoding schemes or element refinements.
Default content type for all elements is xs:string with xml:lang
attribute available.
</xs:documentation>
  <xs:appinfo>
    <rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  <rdf:Description rdf:about="">
    <dc:title>Simple DC XML Schema</dc:title>
    <dc:date>2002-03-12</dc:date>
    <dc:description>This schema defines terms for Simple Dublin
Core, i.e. the 15
elements from the http://purl.org/dc/elements/1.1/ namespace, with
no use of encoding schemes or element refinements.
Default content type for all elements is xs:string with xml:lang
attribute available.</dc:description>
    <dc:publisher>The Dublin Core Metadata Initiative</dc:publisher>
    <dc:creator>Pete Johnston (p.johnston@ukoln.ac.uk)</dc:creator>
    <dc:creator>Carl Lagoze (lagoze@cs.cornell.edu)</dc:creator>
    <dc:creator>Andy Powell (a.powell@ukoln.ac.uk)</dc:creator>
    <dc:creator>Herbert Van de Sompel
(hvdsomp@yahoo.com)</dc:creator>
    <!-- dc:relation
rdf:resource="http://purl.org/dc/elements/1.1/"-->
  </rdf:Description>
</rdf:RDF>
</xs:appinfo>
</xs:annotation>
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
<xs:element name="title" type="elementType"/>
<xs:element name="creator" type="elementType"/>
<xs:element name="subject" type="elementType"/>
<xs:element name="description" type="elementType"/>
<xs:element name="publisher" type="elementType"/>
<xs:element name="contributor" type="elementType"/>
<xs:element name="date" type="elementType"/>
<xs:element name="type" type="elementType"/>
<xs:element name="format" type="elementType"/>
<xs:element name="identifier" type="elementType"/>
<xs:element name="source" type="elementType"/>
<xs:element name="language" type="elementType"/>
<xs:element name="relation" type="elementType"/>
<xs:element name="coverage" type="elementType"/>
<xs:element name="rights" type="elementType"/>
<xs:group name="elementsGroup">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded" minOccurs="0">
      <xs:element ref="title"/>
      <xs:element ref="creator"/>
      <xs:element ref="subject"/>
      <xs:element ref="description"/>
      <xs:element ref="publisher"/>
      <xs:element ref="contributor"/>
      <xs:element ref="date"/>
      <xs:element ref="type"/>
      <xs:element ref="format"/>
      <xs:element ref="identifier"/>
      <xs:element ref="source"/>
      <xs:element ref="language"/>
      <xs:element ref="relation"/>
      <xs:element ref="coverage"/>
      <xs:element ref="rights"/>
    </xs:choice>
  </xs:sequence>
</xs:group>

```

```
</xs:sequence>
</xs:group>
<xs:complexType name="elementType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <!--xs:attribute ref="xml:lang" use="optional"/-->
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:schema>
</csw:SchemaComponent>
</csw:DescribeRecordResponse>
```

3 Architecture

As mentioned above deegree CSW doesn't contain any direct data access modules. It uses deegree WFS through its API or – in future releases – any OGC WFS its their web interface. So deegree CSW can be seen as a kind of facade on top a WFS offering components for transforming incoming catalogue requests into WFS requests and transforming WFS responses into CSW responses.

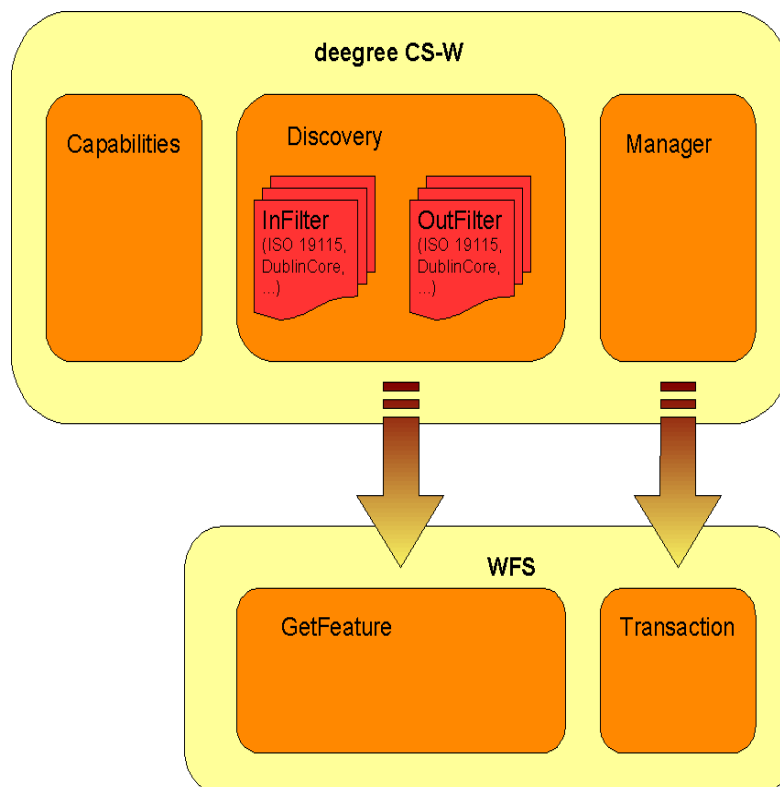


Figure 1: deegree CSW architecture

Figure 1 shows the dependency between deegree CSW and a WFS acting as its datasource. Because transformation modules/scripts are located on the CSW side it is possible to offer several metadata formats/schemas on top of the same WFS FeatureType.

4 Basic configuration

4.1 Structure of the configuration files

The following figure shows the relationships between the different configuration files that have to be adjusted:

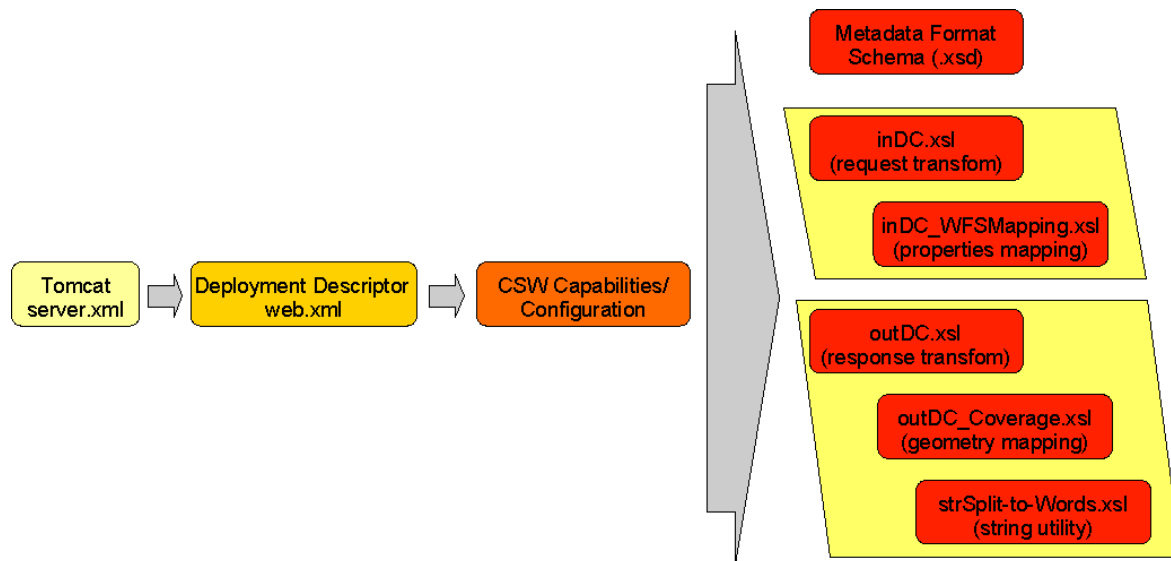


Figure 2 File dependencies for deegree CSW configuration

4.2 The deegree CSW configuration document

The basic configuration allows usage of the full set of parameters that can be used for controlling the deegree CSW. This includes the parameters mentioned in the OGC CSW 2.0.2 specification for the Capabilities document, but also additional technical parameters.

In the following, the elements of the configuration file (an extended Capabilities document) will be described in detail. It is located at `csw_demo/WEB-INF/conf/csw/csw_capabilities.xml`. Appendix A includes an example of a complete configuration/capabilities document.

4.2.1 deegree-Parameters

The `<DefaultOnlineResource>` is the URL by which the CSW operations can be invoked. This parameter can be overwritten by the URLs defined in the request-definitions. Both parameters are mandatory. You have to adjust the `<DefaultOnlineResource>` to your system.

```

<deegree:deegreeParams>
  <deegree:DefaultOnlineResource xlink:type="simple"
    xlink:href="http://localhost:8080/deegree-csw/services"/>
  <deegree:CacheSize>100</deegree:CacheSize>
  <deegree:RequestTimeLimit>60</deegree:RequestTimeLimit>
  <deegree:Encoding>ISO-8859-1</deegree:Encoding>
  <deegree:TransactionInputXSLT xlink:type="simple"
    xlink:href="transaction.xsl"/>
  <deegree:TransactionOutputXSLT xlink:type="simple"
    xlink:href="transactionOut.xsl"/>
  <!-- this parameter will be used to define the default schema used by the
  catalogue if OUTPUTSCHEMA parameter is missing. According to CSW specification this
  must be OGCCORE which is not useful for GetRecordById requests -->
  <deegree:DefaultOutputSchema>csw:profile</deegree:DefaultOutputSchema>
  <deegree:WFSResource xlink:type="simple" xlink:href="wfs_capabilities.xml"/>
</deegree:deegreeParams>

```

The <CacheSize> parameter defines the size of the cache available to deegree-CSW in megabyte. This parameter is optional, its default value is 100 MB. By <RequestTimeLimit> the maximum time span is defined in seconds after which a request has to be processed. If this value is exceeded the processing is cancelled and an exception will be thrown. Its default value is 30 seconds. The encoding element defines the default character encoding to be used within the catalogue if embedded sources doesn't provide encoding informations. This element is optional, default value is 'UTF-8'.

Because deegree CSW uses a WFS as datasource one must define the WFS to be used. This is done by setting the WFSResource element. If it's missing it is assumed that a file named 'wfs_capabilities.xml' is available in the same directory as the catalogue capabilities file. deegree CSW also uses this element to determine if a WFS is 'local' or 'remote'. A 'local' WFS will be connected through the deegree API; a 'remote' WFS uses HTTP connections defined by the OGC WFS 1.0.0 specification. A WFS will be handled as being 'local' if the capabilities document is referenced by a file URL or a relative path. Using a HTTP URL will force treating a WFS as 'remote'.

4.2.2 ServiceIdentification

The ServiceIdentification section is required for giving basic informations about the a CSW to the client/user. This information may be used to manage several catalogues within one client or to store a catalogue metadata in another catalogue. For details see OGC Catalogue Service 2.0.2 and the OWS Common specification.

```

<ows:ServiceIdentification>
  <ows:ServiceType>CSW</ows:ServiceType>
  <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
  <ows:Title>deegree v2.2 demo CSW</ows:Title>
  <ows:Abstract>abstract</ows:Abstract>
  <ows:Keywords>
    <ows:Keyword>deegree</ows:Keyword>
  </ows:Keywords>
  <ows:Fees>NO FEES - IT'S FREE</ows:Fees>
  <ows:AccessConstraints>NONE</ows:AccessConstraints>

```

```
</ows:ServiceIdentification>
```

4.2.3 ServiceProvider

This section provides informations about the person(s) and organisation(s) making a catalogue available. For details see OGC Catalogue Service 2.0.2 and the OWS Common specification.

4.2.4 OperationsMetadata

The OperationsMetadata sections provides information about each operation offered by a catalogue. Mainly these are the names of the operations and the parameters that have to be passed to them. Each parameter can be assigned to a list of possible values. At the moment deegree CSW supports GetCapabilities, GetRecords, DescribeRecord, GetRecordById, Transaction and Harvest operations.

For the parameter 'outputSchema' of the GetRecords operation deegree introduces two additional attributes.

```
<ows:Parameter name="outputSchema">
  <ows:Value deegree:input="xslt/inDC.xsl"
deegree:output="xslt/outDC.xsl">DublinCore</ows:Value>
  <ows:Value deegree:input="xslt/inISO.xsl"
deegree:output="xslt/outISO.xsl">http://www.isotc211.org/2005/gmd</ows:Value>
  <ows:Value deegree:input="xslt/inISO.xsl"
deegree:output="xslt/outISO.xsl">http://www.opengis.net/cat/csw/2.0.2</ows:Value>
</ows:Parameter>
```

The first additional attribute 'deegree:input' references a XSLT script that will be used to transform an incoming GetRecord request into a GetFeature request that will be understood by the underlying WFS. Vice versa the script referenced by 'deegree:output' transforms the feature collection returned from the WFS into a XML document that validates against the schema referenced by 'deegree:schema' attribute of the typeName parameter.

```
<ows:Parameter name="typeName">
  <ows:Value deegree:schema="schema/dc_dataset.xsd">csw:Record</ows:Value>
  <ows:Value
deegree:schema="schema/iso/19139/20060504/gmd/metadataEntity.xsd">gmd:MD_Metadata</ows:Value>
  <ows:Value
deegree:schema="schema/iso/19139/20060504/srv/serviceMetadata.xsd">csw:service</ows:Value>
</ows:Parameter>
```

This always should be referenced through an HTTP URL even if a relative path or a FILE URL is valid too. This schema will also be returned when performing a DescribeRecord request against a type. Details about how to construct the in- and out-scripts will be discussed below.

For general information on how to define service operations in a catalogues capabilities document see OGC Catalogue Service 2.0.2 and OWS Common specification.

4.2.5 Filter_Capabilities

The last section of the capabilities document contains a description of the capabilities of the filter functions offered by a catalogue. You can leave this as it comes with the demo installation because this represents the current capabilities of the deegree Filter Encoding implementation. For details see OGC Catalogue Service 2.0.2, OWS Common and Filter Encoding specification.

5 Advanced configuration

5.1 Manual Tomcat integration

The location of deegree's libraries and the central deegree configuration file `csf_capabilities.xml` should be registered with the Servlet Engine (in this case Apache Tomcat 5.5). Tomcat offers several possibilities to register and configure web contexts.

The easiest way to register deegree web services with Tomcat is to copy the `deegree-csw.war` file to the `$TOMCAT_HOME$/webapps` directory. You can do this either with running or stopped tomcat. If the tomcat is started afterwards, the application should be automatically deployed. Tomcat will unpack the `deegree-wfs.war` file (which is nothing more than a .zip file) to the `webapps` directory automatically. The name of the .war sets the name of the service address:

`http://localhost:8080/deegree-csw`

If you want to do the Tomcat installation process manually use the steps described in the following.

Unpack the `deegree-csw.war` to a directory (e.g. `c:/deegree/webapps/deegree-csw`) of your choice.

Afterwards Tomcat needs information about the root directory of the WFS. The easiest way is to create a XML-file in the directory `$TOMCAT_HOME$/conf/Catalina/localhost`, named the same as the service e.g. `deegree-wfs.xml`, and fill it with the following information

```
<Context docBase="c:/deegree/webapps/deegree-csw" path="/deegree-csw">
</Context>
```

where the `docBase` attribute reflects the physical location of the deegree service in the file system and the `path` attribute describes the virtual location of the main directory of the deegree web service. In the example, the root directory of the service is accessible at `http://my.server.domain/deegree-csw/`. For further information have a look at the Tomcat documentation included with the installation.

The name of the deegree-service directory is arbitrary whereas Tomcat definitely looks for a subdirectory `WEB-INF` (in capital letters – even on a Windows system) in the root directory. You will find this directory after tomcat automatically unpacked the war archive. Here the `Deployment-Descriptor` (`web.xml`) is located, which is analysed by Tomcat to identify the servlet(s) belonging to the application, their names, the parameters that are delivered to the servlet(s) and information about the existing access restrictions.

Before starting deegree WFS the following dataset entry in `web.xml` is essential:

```

<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <description>deegree 2.2 - CSW</description>
  <filter>
    <filter-name>SOAPFilter</filter-name>
    <filter-class>org.deegree.enterprise.servlet.SOAPFacadeServletFilter</filter-
class>
  </filter>
  <servlet>
    <servlet-name>services</servlet-name>
    <servlet-class>org.deegree.enterprise.servlet.OGCServletController</servlet-
class>
    <init-param>
      <param-name>services</param-name>
      <param-value>csw</param-value>
    </init-param>
    <!-- CSW INITIALIZING PARAMETERS -->
    <init-param>
      <param-name>csw.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.CSWHandler</param-value>
    </init-param>
    <init-param>
      <param-name>csw.config</param-name>
      <param-value>WEB-INF/conf/csw/csw_capabilities.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <filter-mapping>
    <filter-name>SOAPFilter</filter-name>
    <url-pattern>/services</url-pattern>
  </filter-mapping>
  <servlet-mapping>
    <servlet-name>services</servlet-name>
    <url-pattern>/services</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>
</web-app>

```

The name of the servlet and of the java-class representing the servlet should be indicated in the <servlet> tags. The servlet-name can be user defined, but care should be taken that the same name that is defined here is also used in the servlet-mapping. The servlet is located in the deegree2.jar library.

The tag <init-param> defines parameters that are analyzed by the servlet, while initializing. The transferred parameters are

- 'services': The value of this parameter contains a comma separated list of OWS that will be made available through the context. In the example only a 'csw' is defined to be available (other possible values at the moment are: wfs, wcs, sos, wpvS and csw).
- For each service listed in the 'service' init-param a handler class and a configuration file must be referenced.
- The name of the init-param for defining the handler starts with the service name ('csw' in the example) followed by '.handler'. The value of this parameter is the name of the handler class to be used. It is possible to

write a different class for this and reference it accordingly. As default 'org.degree.enterprise.servlet.CSWHandler' should be used.

- The name of the init-param for defining the main configuration file of a service also starts with the service name followed by '.config'. Notice that you can use a relative path to the configuration file starting at the WEB-INF directory of the context.

If you want to make more than one service available through a servlet context, web.xml looks like this (the example defines a 'csw' as well as a 'wfs'):

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <description>degree 2.2 - CSW</description>
  <filter>
    <filter-name>SOAPFilter</filter-name>
    <filter-class>org.degree.enterprise.servlet.SOAPFacadeServletFilter</filter-
class>
  </filter>
  <servlet>
    <servlet-name>services</servlet-name>
    <servlet-class>org.degree.enterprise.servlet.OGCServletController</servlet-
class>
    <init-param>
      <param-name>services</param-name>
      <param-value>csw,wfs</param-value>
    </init-param>
    <!-- CSW INITIALIZING PARAMETERS -->
    <init-param>
      <param-name>csw.handler</param-name>
      <param-value>org.degree.enterprise.servlet.CSWHandler</param-value>
    </init-param>
    <init-param>
      <param-name>csw.config</param-name>
      <param-value>WEB-INF/conf/csw/csw_capabilities.xml</param-value>
    </init-param>
    <init-param>
      <param-name>wfs.handler</param-name>
      <param-value>org.degree.enterprise.servlet.WFSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wfs.config</param-name>
      <param-value>WEB-INF/conf/csw/wfs_capabilities.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <filter-mapping>
    <filter-name>SOAPFilter</filter-name>
    <url-pattern>/services</url-pattern>
  </filter-mapping>
  <servlet-mapping>
    <servlet-name>services</servlet-name>
    <url-pattern>/services</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

In case you wish to run more than on deegree service, it is recommended to use the deegree-wms.war as this configuration already includes the services wfs, wcs and wms.

The tag `<servlet-mapping>` defines the alias name for the servlet. It is not necessary that the `<servlet-name>` and `<url-pattern>` are identical. `<url-pattern>` is the name for the parameter the servlet will be called through (part of the base-URL of the service that all requests have to use). Combined with the path settings in `$TOMCAT_HOME$/conf/Catalina/localhost/deegree-csw.xml` and respectively the name of the .war which you deployed in the `$TOMCAT_HOME$/webapps` (in our example deegree-csw) you should be able to point to your WFS (OWS) via the following URL: `http://my.server.domain/deegree-csw/services?`

5.2 Setting up the Harvester

deegree CS-W supports harvesting capability as defined by the ISO APP extended by the possibility to harvest entire catalogues. The installation and configuration of this harvesting capability is described in the following.

5.2.1 Setting up the database

Before the harvester can be activated, some additional tables have to be added to the CSW database. Execute the `create_harvester.SQL` file (located in `csw_home/WEB-INF/scripts/` on your CSW PostgreSQL/Postgis database. This will create 4 tables that can be used for managing the harvesting jobs, and to keep track of the metadata records belonging to the according CSW.

Afterwards you have to point the harvester to its database tables. Therefore, you have to adapt the properties file in the `deegree2.jar` in the following way:

1. Unzip the `WEB-INF/lib/deegree2.jar` in a temporary directory
2. Create a directory named `csw_home/WEB-INF/classes/org/deegree/ogcwebservices/csw/manager`
3. Copy the file `org/deegree/ogcwebservices/csw/manager/harvestrepository.properties` (from the unzipped jar) to `csw_home/WEB-INF/classes/org/deegree/ogcwebservices/csw/manager` and adapt the database connection there.

5.2.2 Activating harvesting functionality

Please note: For activating the harvesting functionality, you have to receive the corresponding files from the deegree 2 SVN repository (svn.wald.intevation.org/deegree/base/trunk). For further information, please contact the deegree users mailing list

(<http://lists.sourceforge.net/lists/listinfo/deegree-users>).

In order to activate the harvesting functionality for the CSW, you have to use the deployment descriptor WEB-INF/web_harvester.xml and rename it to web.xml (Be careful: make a backup copy of the original web.xml file).

The major difference to a deployment descriptor of a none harvesting CSW are following entries:

```
<web-app>
  <context-param>
    <param-name>CSW.config</param-name>
    <param-value>
      /WEB-INF/conf/csw/csw_capabilities_harvester.xml
    </param-value>
  </context-param>
  <listener>
    <listener-class>
      org.deegree.enterprise.servlet.CSWHarvestingContextListener
    </listener-class>
  </listener>
  ...
```

These parameters force the assigned tomcat context to use a special listener that is called during initialization and shutdown of the catalogue service.

Because the harvester informs the “contractor” of the harvesting operations via eMail, you have to configure a SMTP connection for the catalogue service. Therefore, you have to set the following property for tomcat: “-DmailHost=SERVERNAME”. How this has to be done depends on the way how you have installed your Tomcat and how you start it:

a) Tomcat is installed/started as service on a Windows OS: Open the Apache Tomcat configuration console and add the mailHost parameter to the Java Options:

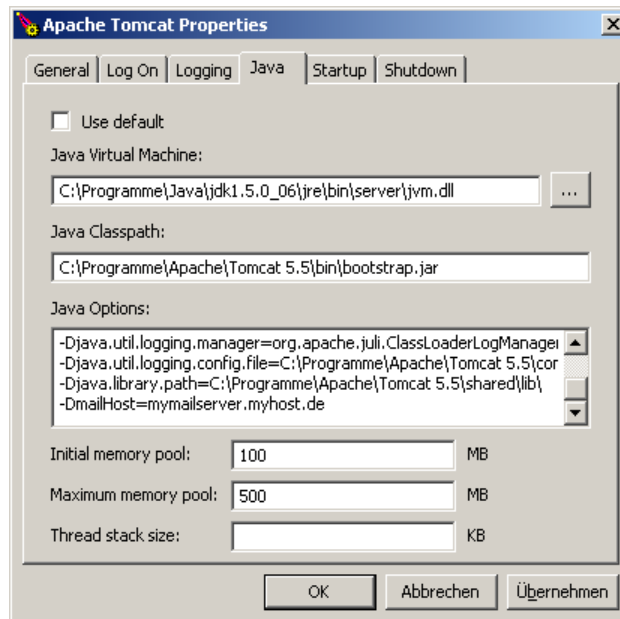


Figure 3: Tomcat console on a Windows OS

b) Tomcat is installed/started as service on a Linux/Unix OS: Set JAVA_OPTS in \$TOMCAT_HOME\$/bin/catalina.sh

```
export JAVA_OPTS="-Xmx500m -Xms100m -DmailHost=SERVERNAME"
```

c) Tomcat ist started directly (e.g using a batch file): Set the option in the Java call:

```
java.exe -Xms100m -Xmx500m -jar -DmailHost=mymailhost.myhost.de
-DCHARSET="UTF-8" -Duser.dir="C:\Programme\Apache\Tomcat 5.5"
"C:\Programme\Apache\Tomcat 5.5\bin\bootstrap.jar" start
```

5.2.3 Testing the harvester

Sending a harvester request – Adding a harvesting job

The following request shows a request that starts harvesting of a full catalog <http://www.google.de/ue>. This functionality is degree-specific and not defined by the ISO APP (only harvesting of single metadata files is defined there).

```
http://my.server.domain/degree-csw/services?
request=Harvest&service=CSW&version=2.0.2&source=http://HOSTNAME:POR
TNUMBER/aCatalogue/services&resourcetype=catalogue&resourceformat=a
pplication/xml&responsehandler=mailto:info@myemail.de&harvestinterva
l=PT2M
```

Important Parameters:	Value	Description

request	<i>Harvest</i>	Fix value
service	<i>CSW</i>	Fix value
version	<i>2.0.2</i>	Fix value
source	http://HOSTNAME:PORTNUMMBER /aCatalogue/services	address of the CSW that is going to be harvested
resourcetype	<i>catalogue</i>	This is a vendor-specific enhancement: Because CSW specification does not specify a way to harvest an entire CSW, deegree uses "catalogue" as parameter.
resourceformat	application/xml	
responsehandler	mailto:info@myemail.de	eMail-address all logging messages go to for this harvest-request
interval	PT2M	Period conforming to ISO8601 Period syntax.
For further information, please consult the OGC CSW 2.0.2 specification.		

Deleting a harvesting job

OGC CSW 2.0 specification does not specify an operation for deleting jobs for harvesting through its interface. In order to delete a harvesting job, you have to follow 2 steps:

1. Delete the job out of the related harvester tables `harvestersource`, `metadatatcache`, `jt_source_responsehandler`, `reponsehandler` within one transaction

The following SQL statements need to be executed step-by-step on the harvester DB:

```
--execute everything in one transaction
begin;

-- find id of the desired harvestsource
```

```
select id from harvestsource where source =
'http://HOSTNAME:PORTNUMMBER/aCatalogue/services';
```

-- find all related id's in responsehandler belonging to the source's id

```
select fk_responsehandler from jt_source_responsehandler where
fk_harvestsource = ID OF DESIRED HARVESTSOURCE;
delete from jt_source_responsehandler where fk_harvestsource =
SOURCE-ID;
delete from responsehandler where id = <LIST OF
fk_harvestsource>;
```

-- figure out all records with their fileidentifiers in the metadacache belonging to the harvestsource

```
select fileidentifier from metadacache where id = ID of
harvestsource;
delete from metadacache where id = ID of harvestsource;
commit;
```

- Using the list of fileidentifiers you have to delete all stored records in the CSW repository through the CSW Transaction Interface via HTTP POST. A request like this is shown in the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Transaction service="CSW" version="2.0.2"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">
  <csw:Delete>
    <csw:Constraint version="1.0.0">
      <ogc:Filter>
        <ogc:PropertyIsLike wildCard="%" singleChar="_" escape="/">
          <ogc:PropertyName>apiso:identifier</ogc:PropertyName>
          <ogc:Literal>FILEIDENTIFIER</ogc:Literal>
        </ogc:PropertyIsLike>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Delete>
</csw:Transaction>
```

Updating the harvestsource parameter

If you want to edit the harvestsource, you first have to delete all according metadata records for that harvestsource-id like described above . Then changing of the harvestsource works like this:

```
update harvestersource set source = 'NEW CATALOGUE URL' where id = 'ID OF THE
SOURCE TO BE CHANGED';
```

Updating the email address to a harvestsource

SQL: execute step by step

```
-- find the accordind foreign keys for the harvestsource in the join table
select fk_responsehandler from jt_source_responsehandler where fk_harvestresource =
ID;
update reponsehandler set address = 'NEW EMAILADDRESS' where id = fk_reponsehandler
and isEmailAddress = 1;
```

Harvesting of one metadata record out of one file:

Send a harvestrequest with a file URL against the service:

```
http://my.server.domain/deegree/services?
request=Harvest&service=CSW&version=2.0.2&source=http://HOSTNAME:POR
TNUMBER/aDirectory/aMetadataFile.xml&resourcetype=http://www.isotc2
11.org/schemas/2005/gmd&resourceformat=application/xml&responsehandl
er=mailto:info@myemail.de&harvestinterval=PT2M
```

Be aware that this file gets harvested once and will remain in the metadata repository until it is deleted manually via the CSW transaction interface (using delete).

5.3 Request and response formatting using XSLT

As mentioned above deegree CSW needs to transform incoming GetRecords requests to WFS GetFeature requests and resulting WFS GML Feature Collections to the requested metadata outputSchema. For this behavior for each supported metadata format/schema two XSLT scripts must be defined, one for input/requests and one for output/responses. The following example explains what those scripts must do. Other than in the demo release where the ISO 19115 schema is configured the next section describes the principally same DublinCore schema as supported metadata format and a deegree WFS serving a feature type named 'DublinCoreProduct' having the following GML application schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.lat-lon.de"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:ll="http://www.lat-lon.de"
elementFormDefault="qualified">
  <!-- import GML 2.1.2 schema definitions -->
  <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="./feature.xsd"/>
  <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="./geometry.xsd"/>
  <!-- ===== -->
  <!-- element definition -->
  <!-- ===== -->
  <xs:element name="FeatureCollection" type="ll:DCFeatureCollectionType"
substitutionGroup="gml:_FeatureCollection"/>
  <xs:complexType name="csw:Record">
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
```

```

<xs:sequence>
  <xs:element name="csw:DUBLINCORE.IDENTIFIER" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.CONTRIBUTOR" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.CREATOR" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.PUBLISHER" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.SUBJECT" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.DESCRPTION" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.RELATION" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.SOURCE" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.RIGHTS" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.FORMAT" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.TYPE" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.TITLE" type="xs:string"/>
  <xs:element name="csw:DUBLINCORE.DATE" type="xs:date" minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.LANGUAGE" type="xs:string"
    minOccurs="0"/>
  <xs:element name="csw:DUBLINCORE.COVERAGE"
    type="gml:GeometryPropertyType" minOccurs="0"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="DCFeatureCollectionType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureCollectionType">
      <xs:sequence/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
</xs:schema>

```

For details about how to configure a deegree WFS please have a look to deegree WFS demo and documentation.

5.3.1 Input/request

According to OGC CSW specification a valid GetRecord request asking for all metadata objects having a subject that equals the literal 'geoscientificInformation' looks like this:

```

<?xml version="1.0" encoding="UTF-8"?>
<csw:GetRecords service="CSW" version="2.0.2" outputFormat="application/xml"
outputSchema="http://www.opengis.net/cat/csw/2.0.2" resultType="RESULTS"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:apiso="http://www.opengis.net/cat/csw/apiso/1.0">
  <csw:Query typeNames="gmd:MD_Metadata">
    <csw:ElementSetName>brief</csw:ElementSetName>
    <csw:Constraint version="1.1.0">
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>apiso:subject</ogc:PropertyName>
          <ogc:Literal>geoscientificInformation</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Query>
</csw:GetRecords>

```

```

    </ogc:Filter>
  </csw:Constraint>
</csw:Query>
</csw:GetRecords>

```

The respective GetFeature request to be performed against the underlying WFS (see schema above) must look like (shorted version):

```

<?xml version="1.0" encoding="UTF-8"?>
<wfs:GetFeature xmlns:wfs="http://www.opengis.net/wfs"
xmlns:app="http://www.deegree.org/app" xmlns:csw="http://www.opengis.net/cat/csw"
xmlns:gml="http://www.opengis.net/gml" xmlns:java="java"
xmlns:mapping="org.deegree.ogcwebservices.csw.iso_profile.Mapping2_0_2"
xmlns:ogc="http://www.opengis.net/ogc" maxFeatures="10" outputFormat="text/xml;
subtype=gml/3.1.1" resultType="hits" startPosition="1">
  <wfs:Query typeName="app:MD_Metadata">
    <wfs:PropertyName>app:fileIdentifier</wfs:PropertyName>
    <wfs:PropertyName>app:hierarchyLevelCode</wfs:PropertyName>
    (...)
    <ogc:Filter>
      <ogc:PropertyIsLike escape="\ " singleChar="?" wildcard="%">
        <ogc:PropertyName>app:MD_Metadata/app:commonQueryableProperties/app:CQP_Main/app:subject</ogc:PropertyName>
        <ogc:Literal>%|geoscientificInformation|</ogc:Literal>
      </ogc:PropertyIsLike>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>

```

If you have a close look to the the GetRecords and the GetFeature request above you see that their structure is more or less identical. The typeName of the GetFeature Query must be determined from the GetRecords outputSchema and the typeName of the GetRecords request.

The transformation of the filter expression is a bit tricky. In principle it should be no problem to substitute the value of PropertyName with the name known by the WFS. The problem is that a filter expression is a recursive datastructure with a possible unlimited depth of nested elements (see Filter Encoding specification for details).

The XSLT that enables the required transformation (shorted version) is given below:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:csw="http://www.opengis.net/cat/csw" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:java="java"
xmlns:mapping="org.deegree.ogcwebservices.csw.iso_profile.Mapping2_0_2">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:param name="NSP">a:a</xsl:param>
  <xsl:variable name="map" select="mapping:new( )"/>
  <xsl:template match="csw:DescribeRecord">
    <xsl:copy-of select="."/>
  </xsl:template>
  <xsl:template match="csw:GetRecords">

```

```

<!-- will be used for GetRecords requests -->
<wfs:GetFeature outputFormat="text/xml; subtype=gml/3.1.1"
xmlns:gml="http://www.opengis.net/gml" xmlns:app="http://www.deegree.org/app">
  <xsl:if test="./@maxRecords != ' ' ">
    <xsl:attribute name="maxFeatures"><xsl:value-of
select="./@maxRecords"/></xsl:attribute>
  </xsl:if>
  <xsl:if test="./@startPosition != ' ' ">
    <xsl:attribute name="startPosition"><xsl:value-of
select="./@startPosition"/></xsl:attribute>
  </xsl:if>
  <xsl:if test="./@resultType = 'HITS' ">
    <xsl:attribute name="resultType">hits</xsl:attribute>
  </xsl:if>
  <xsl:if test="./@resultType = 'RESULTS' ">
    <xsl:attribute name="resultType">results</xsl:attribute>
  </xsl:if>
  <xsl:for-each select="./csw:Query">
    <wfs:Query>
      <xsl:attribute name="typeName">
        app:MD_Metadata
      </xsl:attribute>
      <xsl:apply-templates select="."/>
    </wfs:Query>
  </xsl:for-each>
</wfs:GetFeature>
<xsl:apply-templates select="csw:ResponseHandler"/>
</xsl:template>
<xsl:template match="csw:ResponseHandler"/>
<xsl:template match="csw:Query">
  <xsl:apply-templates select="csw:ElementSetName"/>
  <xsl:for-each select="./child::*">
    <xsl:if test="local-name(.) = 'ElementName' ">
      <wfs:PropertyName>
        <xsl:apply-templates select="."/>
      </wfs:PropertyName>
    </xsl:if>
  </xsl:for-each>
  <xsl:apply-templates select="csw:Constraint"/>
  <xsl:apply-templates select="ogc:SortBy"/>
</xsl:template>
<xsl:template match="ogc:SortBy">
  <ogc:SortBy>
    <xsl:for-each select="./child::*">
      <ogc:SortProperty>
        <ogc:PropertyName>
          <xsl:value-of select="mapping:mapSortProperty( $map,
./ogc:PropertyName, $NSP )"/>
        </ogc:PropertyName>
        <xsl:copy-of select="ogc:SortOrder"/>
      </ogc:SortProperty>
    </xsl:for-each>
  </ogc:SortBy>
</xsl:template>
<xsl:template match="csw:ElementSetName">
  <xsl:if test=". = 'brief' ">
    <wfs:PropertyName>app:fileIdentifier</wfs:PropertyName>
    <wfs:PropertyName>app:hierarchyLevelCode</wfs:PropertyName>
    (..)
  </xsl:if>
  <xsl:if test=". = 'summary' ">
    <wfs:PropertyName>app:fileIdentifier</wfs:PropertyName>
    <wfs:PropertyName>app:language</wfs:PropertyName>
    (..)
  </xsl:if>
  <xsl:if test=". = 'full' "/>
  <xsl:if test=". = 'hits' ">
    <wfs:PropertyName>_COUNT_</wfs:PropertyName>
  </xsl:if>

```



```

</xsl:template>
<xsl:template match="csw:Constraint">
  <ogc:Filter>
    <xsl:apply-templates select="ogc:Filter"/>
  </ogc:Filter>
</xsl:template>
<xsl:template match="ogc:Filter">
  <xsl:apply-templates select="ogc:And"/>
  <xsl:apply-templates select="ogc:Or"/>
  <xsl:apply-templates select="ogc:Not"/>
  <xsl:if test="local-name(./child::*[1]) != 'And' and local-
name(./child::*[1]) != 'Or' and local-name(./child::*[1]) != 'Not'">
    <xsl:for-each select="./child::*">
      <xsl:choose>
        <xsl:when test="local-name(.) = 'PropertyIsEqualTo'">
          <xsl:call-template name="propertyIsEqualTo"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:call-template name="copyProperty"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:for-each>
  </xsl:if>
</xsl:template>
<xsl:template match="ogc:And | ogc:Or | ogc:Not">
  <xsl:copy>
    <xsl:apply-templates select="ogc:And"/>
    <xsl:apply-templates select="ogc:Or"/>
    <xsl:apply-templates select="ogc:Not"/>
    <xsl:for-each select="./child::*">
      <xsl:if test="local-name(.) != 'And' and local-name(.) != 'Or' and local-
name(.) != 'Not'">
        <xsl:choose>
          <xsl:when test="local-name(.) = 'PropertyIsEqualTo'">
            <xsl:call-template name="propertyIsEqualTo"/>
          </xsl:when>
          <xsl:otherwise>
            <xsl:call-template name="copyProperty"/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:if>
    </xsl:for-each>
  </xsl:copy>
</xsl:template>
<xsl:template name="propertyIsEqualTo">
  <xsl:variable name="propName" select="ogc:PropertyName"/>
  <xsl:choose>
    <xsl:when test="contains( $propName, 'subject') or
contains( $propName, 'AlternateTitle') or
contains( $propName, 'ResourceIdentifier') or
contains( $propName, 'ResourceLanguage') or
contains( $propName, 'GeographicDescriptionCode') or
contains( $propName, 'TopicCategory') ">
      <ogc:PropertyIsLike>
        <xsl:attribute name="wildCard">%</xsl:attribute>
        <xsl:attribute name="singleChar">?</xsl:attribute>
        <xsl:attribute name="escape">\</xsl:attribute>
        <xsl:if test="boolean( ./@matchCase )">
          <xsl:attribute name="matchCase"><xsl:value-of
select="./@matchCase"/></xsl:attribute>
        </xsl:if>
        <ogc:PropertyName>
          <xsl:apply-templates select="$propName"/>
        </ogc:PropertyName>
        <xsl:for-each select="./child::*">
          <xsl:if test="local-name(.) = 'Literal'">
            <ogc:Literal>
              <xsl:value-of select="mapping:getLiteralValueIsEqualTo( $map,
. )"/>

```

```

        </ogc:Literal>
    </xsl:if>
    <xsl:if test="local-name(.) != 'PropertyName' and local-name(.) !=
'Literal' ">
        <xsl:copy-of select="."/>
    </xsl:if>
</xsl:for-each>
</ogc:PropertyIsLike>
</xsl:when>
<xsl:otherwise>
    <ogc:PropertyIsEqualTo>
        <xsl:if test="boolean( ./@matchCase )">
            <xsl:attribute name="matchCase"><xsl:value-of
select="./@matchCase"/></xsl:attribute>
        </xsl:if>
        <ogc:PropertyName>
            <xsl:apply-templates select="$propName"/>
        </ogc:PropertyName>
        <xsl:for-each select="./child::*">
            <xsl:if test="local-name(.) != 'PropertyName' ">
                <xsl:copy-of select="."/>
            </xsl:if>
        </xsl:for-each>
    </ogc:PropertyIsEqualTo>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
<xsl:template name="copyProperty">
    <xsl:choose>
        <xsl:when test="local-name(.) = 'PropertyIsLike' ">
            <xsl:copy>
                <xsl:variable name="wildCard" select="./@wildCard"/>
                <xsl:attribute name="wildCard"><xsl:value-of
select="$wildCard"/></xsl:attribute>
                <xsl:attribute name="singleChar"><xsl:value-of
select="./@singleChar"/></xsl:attribute>
                <xsl:attribute name="escape"><xsl:value-of
select="./@escape"/></xsl:attribute>
                <xsl:if test="boolean( ./@matchCase )">
                    <xsl:attribute name="matchCase"><xsl:value-of
select="./@matchCase"/></xsl:attribute>
                </xsl:if>
                <xsl:variable name="propName" select="ogc:PropertyName"/>
                <ogc:PropertyName>
                    <xsl:apply-templates select="$propName"/>
                </ogc:PropertyName>
                <xsl:for-each select="./child::*">
                    <xsl:if test="local-name(.) != 'PropertyName' ">
                        <ogc:Literal>
                            <xsl:value-of select="mapping:getLiteralValueIsLike( $map, ..
$propName, wildCard )"/>
                        </ogc:Literal>
                    </xsl:if>
                </xsl:for-each>
            </xsl:copy>
        </xsl:when>
        <xsl:otherwise>
            <xsl:copy>
                <xsl:if test="boolean( ./@matchCase )">
                    <xsl:attribute name="matchCase"><xsl:value-of
select="./@matchCase"/></xsl:attribute>
                </xsl:if>
                <xsl:variable name="propName" select="ogc:PropertyName"/>
                <ogc:PropertyName>
                    <xsl:apply-templates select="$propName"/>
                </ogc:PropertyName>
                <xsl:for-each select="./child::*">
                    <xsl:if test="local-name(.) != 'PropertyName' ">
                        <xsl:copy-of select="."/>
                    </xsl:if>
                </xsl:for-each>
            </xsl:copy>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

```

```

        </xsl:if>
      </xsl:for-each>
    </xsl:copy>
  </xsl:otherwise>
</xsl:choose>
</xsl:template>
<xsl:template match="ogc:PropertyName | csw:ElementName">
  <xsl:value-of select="mapping:mapPropertyValue( $map, ., $NSP )"/>
</xsl:template>
</xsl:stylesheet>

```

As a matter of course other metadata format/schema like DublinCore (which is currently configured with this demo) or FGDC need special input transformation script. But notice this may map GetRecord requests to the same WFS feature type!

In principle it is possible to use more extensive mappings and to put these mappings into a Java class invoked by the XSLT processor.

Example:

```

<xsl:stylesheet version="1.0" ...
  xmlns:java="java" xmlns:mapping="de.latlon.catalog.Mapping">

<xsl:variable name="HIERARCHY">
...
</xsl:variable>
...
<xsl:template match="ogc:PropertyName | csw:ElementName">
  <!-- mapping property name value -->
  <xsl:value-of select="mapping:mapPropertyValue( ., $HIERARCHY )"/>
</xsl:template>
...

```

5.3.2 output/response

Like the incoming request the response returned from the WFS must be transformed. The request described above will lead to a WFS response that looks like this:

```

<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection numberOfFeatures="1" xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:app="http://www.deegree.org/app"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <gml:boundedBy>
    <gml:Envelope srsName="EPSG:4326">
      <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
      <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <gml:featureMember>
    <app:MD_Metadata gml:id="MD_Metadata_42">
      <gml:boundedBy>
        <gml:Envelope srsName="EPSG:4326">
          <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
          <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <app:contact>
        <app:CI_RespParty gml:id="CI_RespParty_37">
          <app:individualname>Metadata Administrator</app:individualname>

```

```

<app:organisationname>Utah State Government</app:organisationname>
<app:role>
  <app:CI_RoleCode gml:id="CI_RoleCode_7">
    <app:codelistvalue>pointOfContact</app:codelistvalue>
  </app:CI_RoleCode>
</app:role>
<app:contactInfo>
  <app:CI_Contact gml:id="CI_Contact_29">
    <app:address>
      <app:CI_Address gml:id="CI_Address_27">
        <app:deliveryPoint>
          <app:DeliveryPoint gml:id="DeliveryPoint_9">
            <app:deliverypoint>5130 State Office
Building</app:deliverypoint>
          </app:DeliveryPoint>
        </app:deliveryPoint>
        <app:city>Salt Lake City</app:city>
        <app:postalCode>84114</app:postalCode>
        <app:country>USA</app:country>
        <app:electronicMailAddress>
          <app:ElectronicMailAddress
gml:id="ElectronicMailAddress_24">
            <app:email>cclark@utah.gov</app:email>
          </app:ElectronicMailAddress>
        </app:electronicMailAddress>
      </app:CI_Address>
    </app:address>
  </app:CI_Contact>
</app:contactInfo>
</app:CI_RespParty>
</app:contact>
<app:dataIdentification>
  <app:MD_DataIdentification gml:id="MD_DataIdentification_42">
    <gml:boundedBy>
      <gml:Envelope srsName="EPSG:4326">
        <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
        <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
      </gml:Envelope>
    </gml:boundedBy>
    <app:identificationInfo>
      <app:MD_Identification gml:id="MD_Identification_42">
        <app:citation>
          <app:CI_Citation gml:id="CI_Citation_49">
            <app:title>Municipalities 2004 (Archive)</app:title>
            <app:publicationdate>2002-01-
01T00:00:00</app:publicationdate>
            <app:context>Identification</app:context>
          </app:CI_Citation>
        </app:citation>
        <app:abstract><![CDATA[These data were first created through a
cooperative effort of the Governors Office of Planning and Budget, the Utah
Department of Transportation, the Utah State Tax Commission, and AGRC. All
incorporated municipalities in the state through dates varying from 1984 to 1992
were initially represented. Updates for further annexations and deannexations were
completed with the cooperative efforts of the State Tax Commission, Property Tax
Division, and AGRC, and currentness now ranges from 1993 through the end of the
2003 calendar year. Data developed with coordinate geometry (COGO) from legal
descriptions were used. When necessary the data were adjusted to conform to known
physical features. Due to adjustments to resolve conflicting or ambiguous data, the
result was more a physical cadastre than a legal one. In other words, this dataset
does not represent exact legal boundaries as per surveyed description, but, rather
a set of boundaries used for the administrative purposes that conforms to logical
administrative rules (e.g. no two cities or redevelopment areas may cover the same
geogrphic extent).]]></app:abstract>
      </app:MD_Identification>
    </app:identificationInfo>
    <app:topicCategory>
      <app:MD_TopicCategoryCode gml:id="MD_TopicCategoryCode_8">
        <app:category>geoscientificInformation</app:category>
      </app:MD_TopicCategoryCode>
    </app:topicCategory>
  </app:MD_DataIdentification>
</app:dataIdentification>

```

```

    </app:MD_TopicCategoryCode>
  </app:topicCategory>
  <app:boundingBox>
    <app:EX_GeogrBBOX gml:id="EX_GeogrBBOX_40">
      <gml:boundedBy>
        <gml:Envelope srsName="EPSG:4326">
          <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
          <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <app:geom>
        <gml:Surface srsName="EPSG:4326">
          <gml:patches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList srsDimension="2">-114.1 36.98
-114.1 42.01 -109.21 42.01 -109.21 36.98 -114.1 36.98</gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:PolygonPatch>
          </gml:patches>
        </gml:Surface>
      </app:geom>
      <app:queryBBOXOverlap>0.0</app:queryBBOXOverlap>
      <app:crs>EPSG:4326</app:crs>
    </app:EX_GeogrBBOX>
  </app:boundingBox>
  <app:language>en</app:language>
</app:MD_DataIdentification>
</app:dataIdentification>
<app:fileIdentifier>{8E41FF20-873E-44F2-AEF9-
5205E203D41C}</app:fileIdentifier>
<app:dateStamp>2006-12-15T00:00:00</app:dateStamp>
<app:hierarchyLevelCode>
  <app:HierarchyLevelCode gml:id="HierarchyLevelCode_5">
    <app:codelistvalue>dataset</app:codelistvalue>
  </app:HierarchyLevelCode>
</app:hierarchyLevelCode>
<app:commonQueryableProperties>
  <app:CQP_Main gml:id="CQP_Main_42">
    <gml:boundedBy>
      <gml:Envelope srsName="EPSG:4326">
        <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
        <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
      </gml:Envelope>
    </gml:boundedBy>
    <app:subject>|Administration||Ownership||Municipal Boundaries||
Cities||Towns||boundaries||State of Utah||Beaver County||Box Elder County||Cache
County||Carbon County||Daggett County||Davis County||Duchesne County||Emery
County||Garfield County||Grand County||Iron County||Juab County||Kane County||
Millard County||Morgan County||Piute County||Rich County||Salt Lake County||San
Juan County||Sanpete County||Sevier County||Summit County||Tooele County||Uintah
County||Utah County||Wasatch County||Washington County||Wayne County||Weber
County||geoscientificInformation|</app:subject>
    <app:title>Municipalities 2004 (Archive)</app:title>
    <app:abstract><![CDATA[These data were first created through a
cooperative effort of the Governors Office of Planning and Budget, the Utah
Department of Transportation, the Utah State Tax Commission, and AGRC. All
incorporated municipalities in the state through dates varying from 1984 to 1992
were initially represented. Updates for further annexations and deannexations were
completed with the cooperative efforts of the State Tax Commission, Property Tax
Division, and AGRC, and currentness now ranges from 1993 through the end of the
2003 calendar year. Data developed with coordinate geometry (COGO) from legal
descriptions were used. When necessary the data were adjusted to conform to known
physical features. Due to adjustments to resolve conflicting or ambiguous data, the
result was more a physical cadastre than a legal one. In other words, this dataset
does not represent exact legal boundaries as per surveyed description, but, rather
a set of boundaries used for the administrative purposes that conforms to logical

```

```

administrative rules (e.g. no two cities or redevelopment areas may cover the same
geographic extent).]]></app:abstract>
  <app:anyText><![CDATA[Municipalities 2004 (Archive)These data were
first created through a cooperative effort of the Governors Office of Planning and
Budget, the Utah Department of Transportation, the Utah State Tax Commission, and
AGRC. All incorporated municipalities in the state through dates varying from 1984
to 1992 were initially represented. Updates for further annexations and
deannexations were completed with the cooperative efforts of the State Tax
Commission, Property Tax Division, and AGRC, and currentness now ranges from 1993
through the end of the 2003 calendar year. Data developed with coordinate geometry
(COGO) from legal descriptions were used. When necessary the data were adjusted to
conform to known physical features. Due to adjustments to resolve conflicting or
ambiguous data, the result was more a physical cadastre than a legal one. In other
words, this dataset does not represent exact legal boundaries as per surveyed
description, but, rather a set of boundaries used for the administrative purposes
that conforms to logical administrative rules (e.g. no two cities or redevelopment
areas may cover the same geographic extent).(.)]]></app:anyText>
  <app:identifier>{8E41FF20-873E-44F2-AEF9-
5205E203D41C}</app:identifier>
  <app:modified>2006-12-15T00:00:00</app:modified>
  <app:type>dataset</app:type>
  <app:publicationDate>2002-01-01T00:00:00</app:publicationDate>
  <app:resourceIdentifier>|{8E41FF20-873E-44F2-AEF9-5205E203D41C}|
</app:resourceIdentifier>
  <app:resourceLanguage>|en|</app:resourceLanguage>
  <app:bbox>
    <app:CQP_BBOX gml:id="CQP_BBOX_40">
      <gml:boundedBy>
        <gml:Envelope srsName="EPSG:4326">
          <gml:pos srsDimension="2">-114.1 36.98</gml:pos>
          <gml:pos srsDimension="2">-109.21 42.01</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <app:geom>
        <gml:Surface srsName="EPSG:4326">
          <gml:patches>
            <gml:PolygonPatch>
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList srsDimension="2">-114.1 36.98
-114.1 42.01 -109.21 42.01 -109.21 36.98 -114.1 36.98</gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:PolygonPatch>
          </gml:patches>
        </gml:Surface>
      </app:geom>
    </app:CQP_BBOX>
  </app:bbox>
  <app:topicCategory>|geoscientificInformation|</app:topicCategory>
</app:CQP_Main>
  <app:commonQueryableProperties>
</app:MD_Metadata>
</gml:featureMember>
</wfs:FeatureCollection>

```

Transforming the incoming WFS FeatureCollection into a valid GetRecords outputSchema is a bit complicated. The required XSLT - Script is too long to print it right here. It can be found at \$csw_home\$/WEB-INF/conf/csw/xslt/out.ISO.xsl

The problematic part of the transformation is contained within the included XSLT script. The reason for this is, that ISO requires a different format for geographic extent as will the one used by WFS. In our example we use Postgis database for

storing metadata. In our database schema we define a column to take geometries, one for each dataset. Unfortunately a box isn't a geometry so we have to transform the boxes (as extent of the metadatasets) into polygons (having five vertices). Vice versa we must transform the polygon returned from the WFS (if requested) into the box format required for ISO.

If you use other formats for storing geometries (e.g. four number columns containing the four corner coordinates of a metadatasets boundingbox) you have to rewrite the transformation method. Or you may delegate the job to a static method of a java class that will referenced from the XSLT script.

But beside this a bit complicated aspect of offering a metadata format/schema on basis of a WFS feature type it should be clear that it won't be a principle problem to write more than one output transforming script to support more than one metadata format based on the same physical datasource.

Appendix A Example CSW configuration document

Example for a full deegree CSW configuration/capabilities document (including some options that are not supported by deegree CSW yet):

```
<?xml version="1.0" encoding="UTF-8"?>
<csw:Capabilities xmlns:gmd="http://www.isotc211.org/2005/gmd" version="2.0.0"
updateSequence="0" xmlns:ows="http://www.opengis.net/ows"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:csw="http://www.opengis.net/cat/csw"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:deegree="http://www.deegree.org/csw"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <deegree:deegreeParams>
    <deegree:DefaultOnlineResource xlink:type="simple"
xlink:href="http://localhost:8080/deegree-csw/services" />
    <deegree:CacheSize>100</deegree:CacheSize>
    <deegree:RequestTimeLimit>60</deegree:RequestTimeLimit>
    <deegree:Encoding>ISO-8859-1</deegree:Encoding>
    <deegree:TransactionInputXSLT xlink:type="simple"
xlink:href="xslt/transaction.xsl" />
    <deegree:TransactionOutputXSLT xlink:type="simple"
xlink:href="xslt/transactionOut.xsl" />
    <!--
      this parameter will be used to define the default schema used by the
      catalogue if OUTPUTSCHEMA parameter is missing. According to CS-W
      specification this must be OGCCORE which is not useful for GetRecordById
      requests
    -->
    <deegree:DefaultOutputSchema>http://www.isotc211.org/2005/gmd</deegree:DefaultO
utputSchema>
    <deegree:WFSResource xlink:href="LOCALWFS_configuration.xml" />
  </deegree:deegreeParams>
  <ows:ServiceIdentification>
    <ows:ServiceType>CSW</ows:ServiceType>
    <ows:ServiceTypeVersion>2.0.2</ows:ServiceTypeVersion>
    <ows:Title>deegree demo CSW</ows:Title>
    <ows:Abstract>abstract</ows:Abstract>
    <ows:Keywords>
      <ows:Keyword>deegree</ows:Keyword>
    </ows:Keywords>
    <ows:Fees>NO FEES - IT'S FREE</ows:Fees>
    <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider>
    <ows:ProviderName>lat/lon GmbH</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://www.latlon.de" />
    <ows:ServiceContact>
      <ows:IndividualName />
      <ows:PositionName />
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+49 228 18496-0</ows:Voice>
          <ows:Facsimile>+49 228 18496-29</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>Aennchenstr. 19</ows:DeliveryPoint>
          <ows:City>Bonn</ows:City>
          <ows:AdministrativeArea>NRW</ows:AdministrativeArea>
          <ows:PostalCode>53177</ows:PostalCode>
          <ows:Country>Germany</ows:Country>
          <ows:ElectronicMailAddress>info@lat-lon.de</ows:ElectronicMailAddress>
        </ows:Address>
        <ows:OnlineResource xlink:href="http://www.lat-lon.de" />
        <ows:HoursOfService>9am-17pm</ows:HoursOfService>
        <ows:ContactInstructions>personal</ows:ContactInstructions>
      </ows:ContactInfo>
      <ows:Role>PointOfContact</ows:Role>
    </ows:ServiceContact>
  </ows:ServiceProvider>
</csw:Capabilities>
```



```

</ows:ServiceContact>
</ows:ServiceProvider>
<ows:OperationsMetadata>
  <ows:Operation name="GetCapabilities">
    <ows:DCP>
      <ows:HTTP>
        <ows:Get xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services?" />
      </ows:HTTP>
    </ows:DCP>
  </ows:Operation>
  <ows:Operation name="DescribeRecord">
    <ows:DCP>
      <ows:HTTP>
        <ows:Get xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services?" />
        <ows:Post xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services" />
      </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="typeName">
      <ows:Value>csw:Records</ows:Value>
      <ows:Value>gmd:MD_Metadata</ows:Value>
      <ows:Value>csw:service</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="outputFormat">
      <ows:Value>application/xml</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="schemaLanguage">
      <ows:Value>XMLSCHEMA</ows:Value>
    </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="GetRecords">
    <ows:DCP>
      <ows:HTTP>
        <ows:Post xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services" />
      </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="typeName">
      <ows:Value deegree:schema="schema/dc_dataset.xsd">csw:Record</ows:Value>
      <ows:Value
deegree:schema="schema/iso/19139/20060504/gmd/metadataEntity.xsd">gmd:MD_Metadata</
ows:Value>
      <ows:Value
deegree:schema="schema/iso/19139/20060504/srv/serviceMetadata.xsd">csw:service</ows
:Value>
    </ows:Parameter>
    <ows:Parameter name="outputFormat">
      <ows:Value>application/xml</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="outputSchema">
      <ows:Value deegree:input="xslt/inDC.xsl"
deegree:output="xslt/outDC.xsl">DublinCore</ows:Value>
      <ows:Value deegree:input="xslt/inISO.xsl"
deegree:output="xslt/outISO.xsl">http://www.isotc211.org/2005/gmd</ows:Value>
      <ows:Value deegree:input="xslt/inDC.xsl"
deegree:output="xslt/outDC.xsl">http://www.opengis.net/cat/csw/2.0.2</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="resultType">
      <ows:Value>RESULTS</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="ElementSetName">
      <ows:Value>brief</ows:Value>
      <ows:Value>summary</ows:Value>
      <ows:Value>full</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="CONSTRAINTLANGUAGE">

```

```

        <ows:Value>Filter</ows:Value>
    </ows:Parameter>
</ows:Operation>
<ows:Operation name="GetRecordById">
    <ows:DCP>
        <ows:HTTP>
            <ows:Get xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services" />
            <ows:Post xlink:type="simple" xlink:href="http://localhost:8080/deegree-
csw/services" />
        </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="outputSchema">
        <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
    </ows:Parameter>
</ows:Operation>
<ows:Operation name="Transaction">
    <ows:Parameter name="outputSchema">
        <ows:Value deegree:input="xslt/transaction.xsl"
deegree:output="xslt/transactionOut.xsl">http://www.isotc211.org/2005/gmd</ows:Valu
e>
    </ows:Parameter>
    <ows:DCP>
        <ows:HTTP>
            <ows:Post xlink:href="http://localhost:8080/deegree-csw/services" />
        </ows:HTTP>
    </ows:DCP>
</ows:Operation>
<!-- Following configuration element should only be uncommented if your CSW
supports the harvest operation. -->
<!--
    <ows:Operation name="Harvest"> <ows:DCP> <ows:HTTP> <ows:Get
xlink:href="http://localhost:8080/deegree-csw/services?" /> </ows:HTTP> </ows:DCP>
<ows:Parameter
    name="ResourceType">
<ows:Value>http://www.isotc211.org/schemas/2005/gmd</ows:Value>
<ows:Value>catalogue</ows:Value>
    </ows:Parameter></ows:Operation>
-->
<ows:Constraint name="IsoProfiles">
    <ows:Value>http://www.isotc211.org/2005/gmd</ows:Value>
</ows:Constraint>
<ows:Constraint name="PostEncoding">
    <ows:Value>SOAP</ows:Value>
</ows:Constraint>
</ows:OperationsMetadata>
<ogc:Filter_Capabilities>
    <ogc:Spatial_Capabilities>
        <ogc:Spatial_Operators>
            <ogc:BBOX />
            <ogc:Disjoint />
            <ogc:Intersects />
        </ogc:Spatial_Operators>
    </ogc:Spatial_Capabilities>
    <ogc:Scalar_Capabilities>
        <ogc:Logical_Operators />
        <ogc:Comparison_Operators>
            <ogc:Between />
            <ogc:EqualTo />
            <ogc:GreaterThan />
            <ogc:GreaterThanEqualTo />
            <ogc:LessThan />
            <ogc:LessThanEqualTo />
            <ogc:Like />
            <ogc:NotEqualTo />
            <ogc:NullCheck />
        </ogc:Comparison_Operators>
        <ogc:Arithmetic_Operators>
            <ogc:Simple_Arithmetic />

```

```
</ogc:Arithmetic_Operators>  
</ogc:Scalar_Capabilities>  
</ogc:Filter_Capabilities>  
</csw:Capabilities>
```

Appendix B deegree WFS configuration for accessing example metadatasets

WFS capabilities:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_Capabilities xmlns:deegree="http://www.deegree.org/wfs"
xmlns:ows="http://www.opengis.net/ows" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1.0"
updateSequence="0">
  <!-- ===== -->
  <!-- DEEGREE PARAMETERS -->
  <!-- ===== -->
  <deegree:deegreeParams>
    <!-- mandatory; used as the default URL for omitted DCP-elements (in the
OperationMetadata section) -->
    <deegree:DefaultOnlineResource xlink:type="simple"
xlink:href="http://localhost:8080/deegree-csw/services" />
    <!-- optional; default = 100 (MB); cache size available for storing feature
instances in memory,
not implemented yet -->
    <deegree:CacheSize>250</deegree:CacheSize>
    <!-- optional; default = 30 (seconds); maximum time allowed for the execution
of a request -->
    <deegree:RequestTimeLimit>120</deegree:RequestTimeLimit>
    <!-- optional; default = same directory as configuration; list of directories
to be scanned for featuretypes/datastores to be served by the WFS -->
    <deegree:DataDirectoryList>
      <deegree:DataDirectory>featuretypes</deegree:DataDirectory>
    </deegree:DataDirectoryList>
  </deegree:deegreeParams>
  <!-- ===== -->
  <!-- SERVICE IDENTIFICATION SECTION -->
  <!-- ===== -->
  <ows:ServiceIdentification>
    <ows:Title>localwfs</ows:Title>
    <ows:ServiceType>WFS</ows:ServiceType>
    <ows:ServiceTypeVersion>1.1.0</ows:ServiceTypeVersion>
    <ows:Fees>None</ows:Fees>
    <ows:AccessConstraints>None</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <!-- ===== -->
  <!-- SERVICE PROVIDER SECTION -->
  <!-- ===== -->
  <ows:ServiceProvider>
    <ows:ProviderName>lat/lon GmbH</ows:ProviderName>
    <ows:ProviderSite xlink:href="http://www.lat-lon.de" />
    <ows:ServiceContact>
      <ows:IndividualName>Markus Schneider</ows:IndividualName>
      <ows:PositionName>deegree WFS core developer</ows:PositionName>
      <ows:ContactInfo>
        <ows:Phone>
          <ows:Voice>+49 228 184960</ows:Voice>
          <ows:Facsimile>+49 228 1849629</ows:Facsimile>
        </ows:Phone>
        <ows:Address>
          <ows:DeliveryPoint>Aennchenstr. 19</ows:DeliveryPoint>
          <ows:City>Bonn</ows:City>
          <ows:AdministrativeArea>Northrhine-Westfalia</ows:AdministrativeArea>
          <ows:PostalCode>53177</ows:PostalCode>
          <ows:Country>Germany</ows:Country>
          <ows:ElectronicMailAddress>info@lat-lon.de</ows:ElectronicMailAddress>
        </ows:Address>
      </ows:ContactInfo>
    </ows:ServiceContact>
  </ows:ServiceProvider>

```

```

    <ows:OnlineResource xlink:href="http://www.lat-lon.de" />
    <ows:HoursOfService>9am - 4pm (CET)</ows:HoursOfService>
    <ows:ContactInstructions>Email preferred</ows:ContactInstructions>
  </ows:ContactInfo>
  <ows:Role>PointOfContact</ows:Role>
</ows:ServiceContact>
</ows:ServiceProvider>
<!-- ===== -->
<!-- OPERATIONS METADATA SECTION -->
<!-- ===== -->
<ows:OperationsMetadata>
  <ows:Operation name="GetCapabilities">
    <!-- ows:DCP element omitted -> filled automatically with
DefaultOnlineResource value -->
    <ows:Parameter name="AcceptVersions">
      <ows:Value>1.1.0</ows:Value>
      <!-- Just Version 1.1.0 is supported -->
      <!-- <ows:Value>1.0.0</ows:Value-->
    </ows:Parameter>
    <ows:Parameter name="AcceptFormats">
      <ows:Value>text/xml</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="Sections">
      <ows:Value>ServiceIdentification</ows:Value>
      <ows:Value>ServiceProvider</ows:Value>
      <ows:Value>OperationsMetadata</ows:Value>
      <ows:Value>FeatureTypeList</ows:Value>
      <ows:Value>ServesGMLObjectList</ows:Value>
      <ows:Value>SupportsGMLObjectList</ows:Value>
      <ows:Value>Filter_Capabilities</ows:Value>
    </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="DescribeFeatureType">
    <!-- ows:DCP element omitted -> filled automatically with
DefaultOnlineResource value -->
    <ows:Parameter name="outputFormat">
      <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
    </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="GetFeature">
    <!-- ows:DCP element omitted -> filled automatically with
DefaultOnlineResource value -->
    <ows:Parameter name="resultType">
      <ows:Value>results</ows:Value>
      <ows:Value>hits</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="outputFormat">
      <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
    </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="Transaction">
    <!-- ows:DCP element omitted -> filled automatically with
DefaultOnlineResource value -->
    <ows:Parameter name="inputFormat">
      <ows:Value>text/xml; subtype=gml/3.1.1</ows:Value>
    </ows:Parameter>
    <ows:Parameter name="idgen">
      <ows:Value>GenerateNew</ows:Value>
      <ows:Value>UseExisting</ows:Value>
      <!-- <ows:Value>ReplaceDuplicate</ows:Value> -->
    </ows:Parameter>
    <ows:Parameter name="releaseAction">
      <ows:Value>ALL</ows:Value>
    </ows:Parameter>
  </ows:Operation>
  <!-- -->
  <ows:Operation name="LockFeature" />
  <ows:Operation name="GetFeatureWithLock" />

```

```

<!-- Sets default CRS -->
<ows:Parameter name="srsName">
  <ows:Value>EPSG:4326</ows:Value>
</ows:Parameter>
<!-- Maximum number of features, which the WFS will return to the client -->
<ows:Constraint name="DefaultMaxFeatures">
  <ows:Value>200</ows:Value>
</ows:Constraint>
<!-- time in minutes until locked features expire; not evaluated yet -->
<ows:Constraint name="DefaultLockExpiry">
  <ows:Value>5</ows:Value>
</ows:Constraint>
</ows:OperationsMetadata>
<!-- ===== -->
<!-- FEATURE TYPE LIST SECTION -->
<!-- ===== -->
<wfs:FeatureTypeList />
<!-- ===== -->
<!-- FILTER CAPABILITIES SECTION -->
<!-- ===== -->
<ogc:Filter_Capabilities>
  <!-- Don't change <ogc:Filter_Capabilities>, its client information only -->
  <ogc:Spatial_Capabilities>
    <ogc:GeometryOperands>
      <ogc:GeometryOperand>gml:Envelope</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Point</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:LineString</ogc:GeometryOperand>
      <ogc:GeometryOperand>gml:Polygon</ogc:GeometryOperand>
    </ogc:GeometryOperands>
    <ogc:SpatialOperators>
      <ogc:SpatialOperator name="BBOX" />
      <ogc:SpatialOperator name="Equals" />
      <ogc:SpatialOperator name="Disjoint" />
      <ogc:SpatialOperator name="Intersects" />
      <ogc:SpatialOperator name="Touches" />
      <ogc:SpatialOperator name="Crosses" />
      <ogc:SpatialOperator name="Within" />
      <ogc:SpatialOperator name="Contains" />
      <ogc:SpatialOperator name="Overlaps" />
      <ogc:SpatialOperator name="Beyond" />
    </ogc:SpatialOperators>
  </ogc:Spatial_Capabilities>
  <ogc:Scalar_Capabilities>
    <ogc:LogicalOperators />
    <ogc:ComparisonOperators>
      <ogc:ComparisonOperator>LessThan</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>GreaterThan</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>LessThanEqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>GreaterThanEqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>EqualTo</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>Like</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>Between</ogc:ComparisonOperator>
      <ogc:ComparisonOperator>NullCheck</ogc:ComparisonOperator>
    </ogc:ComparisonOperators>
    <ogc:ArithmeticOperators>
      <ogc:SimpleArithmetic />
    </ogc:ArithmeticOperators>
  </ogc:Scalar_Capabilities>
  <ogc:Id_Capabilities>
    <ogc:EID />
    <ogc:FID />
  </ogc:Id_Capabilities>
</ogc:Filter_Capabilities>
</wfs:WFS_Capabilities>

```

Datastore configuration (ISO 19115):

\$csw_home\$/WEB-INF/conf/csw/featuretypes/csw_postgres.xsd

Because of the amount of data the gml application schema can not be shown here.

Datastore configuration (Dublin core):

Not configured with this demos!

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/cat/csw"
  xmlns:gml="http://www.opengis.net/gml" xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wfs="http://www.deegree.org/wfs"
  xmlns:csw="http://www.opengis.net/cat/csw" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import namespace="http://www.opengis.net/gml"
  schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
  <!-- configuration for the persistence backend to be used -->
  <xs:annotation>
    <xs:appinfo>
      <wfs:Prefix>csw</wfs:Prefix>
      <wfs:Backend>POSTGIS</wfs:Backend>
      <wfs:DefaultSRS>EPSG:4326</wfs:DefaultSRS>
      <JDBCConnection xmlns="http://www.deegree.org/jdbc">
        <Driver>org.postgresql.Driver</Driver>
        <Url>jdbc:postgresql://hurricane:5432/csw</Url>
        <User>deegreetest</User>
        <Password>deegreetest</Password>
        <SecurityConstraints/>
        <Encoding>iso-8859-1</Encoding>
      </JDBCConnection>
    </xs:appinfo>
  </xs:annotation>
  <!-- ===== -->
  <xs:element name="Record" type="csw:RecordType"
    substitutionGroup="gml:_Feature"/>
  <!-- ===== -->
  <xs:complexType name="RecordType">
    <xs:annotation>
      <xs:appinfo>
        <wfs:table>dublincore</wfs:table>
        <wfs:gmlId prefix="DC_">
          <wfs:MappingField field="IDENTIFIER" type="VARCHAR"/>
        </wfs:gmlId>
      </xs:appinfo>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractFeatureType">
        <xs:sequence>
          <xs:element name="DUBLINCORE.IDENTIFIER" type="xs:string">
            <xs:annotation>
              <xs:appinfo>
                <wfs:Content>
                  <wfs:MappingField field="identifier"
                    type="VARCHAR"/>
                </wfs:Content>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="DUBLINCORE.CONTRIBUTOR" type="xs:string">
            <xs:annotation>
              <xs:appinfo>
                <wfs:Content>
                  <wfs:MappingField field="contributor"

```

```

        type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.CREATOR" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="creator"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.PUBLISHER" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="publisher"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.SUBJECT" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="subject"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.DESCRPTION" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="abstract"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.RELATION" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="relation"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.SOURCE" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>
        <wfs:MappingField field="source"
          type="VARCHAR"/>
      </wfs:Content>
    </xs:appinfo>
  </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.RIGHTS" type="xs:string">
  <xs:annotation>
    <xs:appinfo>
      <wfs:Content>

```



```

        <wfs:MappingField field="rights"
            type="VARCHAR"/>
    </wfs:Content>
</xs:appinfo>
</xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.FORMAT" type="xs:string">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="format"
                    type="VARCHAR"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.TYPE" type="xs:string">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="type" type="VARCHAR"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.TITLE" type="xs:string">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="title"
                    type="VARCHAR"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.DATE" type="xs:string">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="dcdate"
                    type="VARCHAR"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.LANGUAGE" type="xs:string">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="language"
                    type="VARCHAR"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
<xs:element name="DUBLINCORE.COVERAGE"
    type="gml:GeometryPropertyType">
    <xs:annotation>
        <xs:appinfo>
            <wfs:Content>
                <wfs:MappingField field="geom"
                    type="GEOMETRY"/>
            </wfs:Content>
        </xs:appinfo>
    </xs:annotation>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>

```

```
</xs:complexType>  
<!-- =====>  
</xs:schema>
```