deegree owsProxy
v2.5

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

Datum	***Beschreibung***	***Author***
2006-10-23	Kleinere Layout-Anpassungen	Markus Müller
2006-11-02	Alle Beispiele mit prefixes versehen; WFSPolicy-Beispiel umd FeatureTypes ergänzt	Andreas Poth
2007-01-15	Harmonisierung mit Standard-deegree Dokumentationsstruktur	Markus Müller
2007-03-01	Präszisierung zur Benutzung eines WAS	Markus Müller
2007-03-12	Präzisierung der Condition-Parameter; Ergänzung der WFS-Policy um transaktionale Parameterdefinitionen	Andreas Poth
2007-03-16	Redaktionelle Anpassungen	Markus Müller
2008-01-31	Ergänzung neuer Konfigurationsmöglichkeiten	Andreas Poth
2008-02-04	Rechtschreibekorrekturen; Umstellung des Dokumentes auf deutsche Sprache	Markus Lupp
2008-05-19	Translation into English	Markus Lupp
2008-10-08	Bug fix	Andreas Poth
	Update for 2.3 (to be done)	

Table of Contents

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 is comprised of OGC Web Services implementations as well as clients for these services. It is Open Source Software covered by the GNU Lesser General Public License (GNU LGPL) and is available via the following URL:

`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 concerns the setup and configuration of deegree owsproxy. U3R is part of deegree iGeoSecurity and can be used in conjunction with deegree U3R and deegree WASS.

The specifications of the Open Geospatial Consortium (OGC) define the fundamentals for interoperability in Spatial Data Infrastructures (SDI). As interface specifications they do not define mechanisms for handling of users, access rights and security mechanisms. If the Internet address of a Web Map Service or Catalogue Service is known, it can be used by anybody. Especially since a great number of OWS-Clients is available, access of open interfaces is no expert knowledge any more. Usually the complete data served is freely accessible. Information about available data is according to the specifications a substantial requirement and therefore desirable under all circumstances.

A proxy in front of the actual OGC Web Service (OWS) can behave like the OWS itself, but offer additional possibilities for filtering in- and output, e.g.:

- Several views on the OWS can be created. Different users can only access some of these views (WMS1 for planning data, WMS2 for topography, WMS3 for waters, etc.)
- Inside one owsProxy different users can only access specific Layers, FeatureTypes, etc.
- Access can also be restricted spatially or thematically, e.g. only for a specific quarter of a city, a specific output format or a maximum resolution etc.
- These restrictions can be controlled seperately for the different operations of a OWS (e.g. `GetCapabilities`, `GetMap`, `GetFeature`, `GetRecords`, `Transaction` etc.).
- Access rights can be defined globally as well as user-specific.

The owsProxy that is implemented as part of deegree allows definition of all mentioned possibilities for filtering. Currently, OGC WMS, WFS and CSW can be secured using owsProxy.

Besides owsproxy, deegree comprises a number of additional services and clients. A complete list of deegree components can be found using the following URL and links:

`http://www.lat-lon.de` → Products

Downloads of packaged deegree components can be found via:

`http://www.deegree.org` → Download

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

To run deegree2 owsProxy you need:

- Java (JRE or JSDK) version 1.5.x
- Tomcat 5.5.x

For installation of these components refer to the corresponding documentation at java.sun.com and tomcat.apache.org respectively.

2.2 deegree owsProxy release

Until an installation package (Demo Release) for owsProxy is available, all the corresponding files have to be obtained from the deegree SVN.

To install owsProxy, the following components are needed:

- a configuration for the WAS and/or the WSS (the WAS is optional)
- a deegree2.jar (either precompiled or created out of the classes in the CVS)
- optional: a U3R-Database
- a suitable OGC Web Services

2.3 Installation

deegree owsproxy is implemented as a Java Servlet Filter and not restricted to work only with deegree OWS. If the service to be protected resides on a physically or logically different server, deegree owsProxy can be combined with a simple Proxy-Servlet, that relates incoming, valid requests to the corresponding service and also relates the responses. This allows to use owsProxy to work also with OWS that are not implemented as Java Servlets like for example UMN MapServer.

To install owsProxy has to be bound to a Web Context first of all. This is done by creating the corresponding entries in the Deployment Descriptor of the context. The path to the archive deegree2.jar has to be registered in the classpath of the Servlet Engine.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!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>
   <filter>
      <filter-name>OWSProxy</filter-name>
      <filter-class>
```

```
            org.deegree.security.owsproxy.ConfigurableOWSProxyServletFilter
        </filter-class>
        <init-param>
            <param-name>WMS:POLICY</param-name>
            <param-value>./resources/wmspolicy.xml</param-value>
        </init-param>
        <init-param>
            <param-name>CSW:POLICY</param-name>
            <param-value>./resources/cswpolicy.xml</param-value>
        </init-param>
        <init-param>
        <init-param>
            <param-name>AuthenticationSettings</param-name>
            <param-value>/WEB-INF/conf/security/authentication.xml</param-value>
        </init-param>
        <init-param>
            <param-name>PROXYURL</param-name>
            <param-value>http://myHost:8080/owsproxy/proxy</param-value>
        </init-param>
        <init-param>
            <param-name>ALTREQUESTPAGE</param-name>
            <param-value>/altrequestpage.jsp</param-value>
        </init-param>
        <init-param>
            <param-name>ALTRESPONSEPAGE</param-name>
            <param-value>/altresponsepage.jsp</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>OWSProxy</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <servlet>
        <servlet-name>SimpleProxyServlet</servlet-name>
        <servlet-class>
            org.deegree.enterprise.servlet.SimpleProxyServlet
        </servlet-class>
        <init-param>
            <param-name>WMS:HOST</param-name>
            <param-value>http://localhost:8081/deegree/services</param-value>
        </init-param>
        <init-param>
            <param-name>CSW:HOST</param-name>
            <param-value>http://localhost:8081/deegree/services</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>SimpleProxyServlet</servlet-name>
        <url-pattern>/proxy</url-pattern>
    </servlet-mapping>
</web-app>
```

In this example, owsProxy is **not** installed in the same context as the WMS and CSW services that will be protected. Therefore the Proxy-Servlet has to be used. Each request meant to be sent to deegree WMS or CSW will therefore be received by owsProxy and checked for its validity. Only if the filter identifies the request as valid, it will be forwarded to the Proxy-Servlet that relates the request to the OWS identified by WMS:HOST or CSW:HOST. Responses will also be related, validated and possibly modified by the Servletfilter.

OwsProxy as well as the ProxyServlet need a number of initialisation parameters when they are registered to the Deployment Descriptor. To make known to the owsProxy for which OWS he will act as filter and also to pass the corresponding filter rules, a parameter starting with the OWS type followed by 'Policy' has to be defined. The value of this parameter references a so-called policy file containing all rules that have to be applied. A detailed description of the policies is given in chapter 3.Using the initialisation parameter 'AuthenticationSettings' an XML file is referenced, that defines the order of the mechanism used for authentication of users. This also will be described thoroughly in chapter 3.

If the deegree Proxy Servlet is used, it is necessary to define for which services it will act as Proxy. An instance of the Proxy Servlet can at the same time be used for several, different OWS, but only for one service of a type at once. Using the init parameters it will be defined to which services requests will be related. The corresponding parameter names start with the service type followed by ':HOST'. The value of the parameter is the Base-URL of the corresponding service.

3 Basic configuration

deegree owsProxy implements a pessimistic access concept, i.e. all access (all possible values of known parameters) is denied first of all. An administrator therefore has to allow any access and any allowed values explicitly using so-called policy files. To make this possible in an efficient way, whole value ranges for single parameters or parameter group can, among other things, be made accessible as a whole.

Additionally it is possible to couple access to specific methods are value ranges of single parameters to individual user rights. For this end deegree owsProxy has to be coupled with the User and Rights Administration (U3R).

OwsProxy is equipped with several authentication mechanisms to enable it to relate requests to known users. Which mechanism is actually used is, as described above, defined in a separate XML-file that is passed to owsProxy using the initialisation parameter 'AuthenticationSettings'.

3.1 Determining user authentication

OwsProxy suports four predefined methods to authenticate users, that all use deegree's Rights administration (see U3R documentation):

1. Username/Password: Username and password have to be supplied using the predefined parameters as part of the request.
2. Web Authentication Service (WAS): This is a service specific to the SDI initiative of Northrhine-Westfalia in Germany (GDI-NRW). It delivers a SessionID with a limited lifetime or checks an existing SessionID if it is valid.
3. UserPrincipal: Using this mechanism, the user name is extracted from the HTTP header of the incoming request. Usually the Web Server that received the request already authenticated the user when the request is handed over to owsProxy so that owsProxy can only get the username. Therefore, the users shall not be given passwords in U3R (see also U3R documentation).
4. IP-Address: In this case the IP address of the requesting user is checked using U3R. Exact addresses (e.g. 127.0.0.1) as well as address patterns (e.g. 19.1.10.*) can be used this way. Formally such an address or pattern relates to a username in U3R and therefore has to be defined ther (withou password).

The authentication mechanisms used by owsProxy are, as already stated, defined in a separate file. This includes the authentication methods known to a owsProxy embedded in the root element.

```
<?xml version="1.0" encoding="UTF-8"?>
<Authentications>
   <Method name="user:password">
      <class>org.deegree.security.UserPasswordAuthentication</class>
   </Method>
   <Method name="WAS">
      <class>org.deegree.security.WASAuthentication</class>
      <init-param>
         <name>WAS</name>
         <value>
<![CDATA[http://localhost:8081/was/was?
REQUEST=DescribeUser&Service=WAS&version=1.0.0&SESSIONID=[SESSIONID]]]>
         </value>
      </init-param>
   </Method>
   <Method name="UserPrincipal">
      <class>org.deegree.security.UserPrincipalAuthentication</class>
   </Method>
   <Method name="IP-Address">
      <class>org.deegree.security.IPAddressAuthentication</class>
      <init-param>
         <name>pattern</name>
         <value>127.0.0.*,localhost,19.1.10.21</value>
      </init-param>
   </Method>
</Authentications>
```

As can be seen in the above example, such a file can define several authentication mechanisms. In this case owsProxy first tries if a user can be authenticated using the first defined mechanism. If this is not possible, the second one is tried. This is continued until either the authentication is successful or all defined methods were applied unsuccessfully. In the last case the request of the user is refused.

For each authentication method the name of the responsible Java class as well as the needed initialisation parameters are supplied. The used Java classes have to implement the abstract class org.deegree.security.AbstractAuthentication. It is therefore possible to declare further authentication methods without interfering with the existing deegree code.

3.1.1 User name / password

This is the most simple of the authentication mechanisms in deegree. For this, user name and password are extracted from an incoming request and validated against information stored in deegree U3R. This authentication mechanism is declared by setting the attribute Method/@name='user:password' (see example above).

In the case of KVP-encoded requests (usually via HTTP-Get) user name and password are extracted from the parameters 'USER' and 'PASSWORD'. When using XML-encoded requests (usually via HTTP-Post) it is expected that the root element includes the attributes 'user' and 'password'.

3.1.2 Web Authentication Service

The Web Authentication Service (WAS) was developed as part of the GDI-NRW intiative. This authentication method that is declared with Method/@name='WAS' uses a WAS to validate a SessionID that has to be provided inside a request. After validating the SessionID the corresponding user is identified. For this end, owsProxy need a request pattern inside which a place holder [SESSIONID] is replaced by the SessionID provided with the request. This pattern is passed to the authentication method via the initialisation parameter 'WAS' (details regarding WAS can be found in the corresponding GDI-NRW specification as well as in the deegree WASS documentation.

3.1.3 UserPrincipal

Especially inside of networks where users are logged in at a central node, user authentication when sending requests against servers is sent inside the HTTP-header. A request, respectively the login value inside the header, is in this case first checked by the server before the request is passed on to the owsProxy. Because of this, only the user name is available to owsProxy, as she is already authenticated against the network respectively the server, no additional check against information stored in U3R is necessary. If UserPrincipal is to be used as authentication method, the corresponding users have to be stored without password in U3R!

If a incoming request does not include a login value in the header, deegree uses the default value that is stored in the file org.deegree.enterprise.servlet.ServletRequestWrapper.properties. In this case the user acts as anonymous. ***Because of this fact, the authentication method 'UserPrincipal' always has to be defined when anonymous users are given access rights in U3R!*** If access rights are declard in the policy files described later, the declaration of the authentication method 'WAS' is not necessary.

3.1.4 IP-Address

The last of deegree's predefined authentication methods uses the IP-address of a requesting user to authenticate against U3R. For this, IP-address and corresponding patterns can be handled by deegree U3R just like normal users (without password).

OwsProxy checks in the case of usage the method declared by Method/name='IP-Address', if the address of a requesting user corresponds to one of the comma-separated address/pattern, that are passed using the initialisation parameter 'pattern' to the authentication method. If this is the case, the address respectively the pattern is used as user name in the following. This means that the check if the existing rights are sufficient for execution of the request is done using a user whose name corresponds to the matching IP-pattern respectively IP-address.

By using this method it is possible to allow all members of an organisation like a private company or a public agency to authenticate using their IP-address. So, all access rights can be assigned once for all members of the organisation and don't have to be defined separately.

3.2 Policy file

A policy file includes information about allowed access to a service i.e. the information defined here does not define refusals. Furthermore, in a policy file it is possible to define if a coupling to U3R shall be used and, if yes, which parameters are affected by this. Because of the different functions and parameters the policies for different service types are also quite different. Only a small number of elements is used by all service types. The following example shows an excerpt from a policy file as it is common to all service types:

```
<?xml version="1.0" encoding="UTF-8"?>
<OWSPolicy xmlns="http://www.deegree.org/security" service="CSW">
   <Security>
      <RegistryClass>org.deegree.security.drm.SQLRegistry</RegistryClass>
      <ReadWriteTimeout>300</ReadWriteTimeout>
      <RegistryConfig>
         <jdbc:JDBCConnection xmlns:jdbc="http://www.deegree.org/jdbc">
            <jdbc:Driver>String</jdbc:Driver>
            <jdbc:Url>String</jdbc:Url>
            <jdbc:User>String</jdbc:User>
            <jdbc:Password>String</jdbc:Password>
            <jdbc:SecurityConstraints/>
            <jdbc:Encoding>ISO-8859-1</jdbc:Encoding>
         </jdbc:JDBCConnection>
      </RegistryConfig>
      <authentificationSettings>
         <authentificationService>
            <OnlineResource
               xlink:type="simple" xlink:href="http://localhost:8081/was/was"/>
         </authentificationService>
      </authentificationSettings>
   </Security>
   <GeneralConditions>
      <Conditions>
         <Parameter name="getContentLength" userCoupled="false">
            <Value>1000</Value>
         </Parameter>
         <Parameter name="postContentLength" userCoupled="false">
            <Value>1000000</Value>
         </Parameter>
         <Parameter name="header" userCoupled="false">
            <Any/>
         </Parameter>
```

```
        <Parameter name="requestMethod" userCoupled="false">
           <Value>GET,POST</Value>
        </Parameter>
      </Conditions>
   </GeneralConditions>
   <Requests>
      [...]
   </Requests>
</OWSPolicy>
```

In the <Security>-element inside the root tag the optional definition of the access information to U3R and the (also optional) authentication service can be given. The block as a whole is only needed, when some or all request parameters should be coupled to individual user rights. After the security-block the definition of general, i.e. not service-specific access rights can be given. These are stored in <GeneralConditions> and refer to selected HTTP-parameters.

3.2.1 Security

If a coupling to individual access rights is defined, owsProxy needs information about the type of registry is intended to be used (currently, only org.deegree.security.drm.SQLRegistry is available) and about access to it. A SQLRegistry is realised by a database schema, so the needed access parameters have to be given for the corresponding database.

- Driver: Java driver class for ODBC access (depends on the used database, in the example an ODBC database is configured).
- Url: address and name of the corresponding database that the database can be accessed at via JDBC.
- User: User name for accessing the database (optional)
- Password: (optional)

The element <ReadWriteTimeout> defines after how many milliseconds the access administration cancels the attempt to access a registry, if this does not respond.

Besides information about accessing a User- and Access rights database, the address of a authentication service can also be defined inside the security element. This information is optional, if it is not provided, a user has to supply his user name (NAME=XYZ) and password (PASSWORD=ABC) with each sent request. If on the other hand an authentication service is used, a sessionID (e.g. SESSIONID=2e3r45t) can be used, which is much more secure. The users has to get this sessionID beforehand by using the GetSession operation of an authentication service. In this case, owsProxy first checks if the sessionID is valid and the determines the corresponding user information that allows validation of the request against U3R. If neither a sessionID nor username/password are supplied, owsProxy assumes an anonymous user. The login of an anonymous user can be configured using org.deegree.enterprise.servlet.ServletRequestWrapper.properties. The default behaviour this is set to 'default'.

Example.:

```
http://myhost:8080/deegree/proxy?
request=GetCapabilities&version=1.1.1&service=WMS&&...&user=latlon&p
assword=latlonspassword


http://myhost:8080/deegree/proxy?
request=GetMap&version=1.1.1&bbox=1,1,10,10&width=400&height=500  ...
&sessionID=3tr364g2
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<GetFeature outputFormat="GML2" xmlns="http://www.opengis.net/wfs"
 xmlns:gml="http://www.opengis.net/gml" xmlns:ogc="http://www.opengis.net/ogc"
 sessionID="5z4g447">
   <Query typeName="Europe">
      <ogc:Filter>
         <ogc:BBOX>
            <ogc:PropertyName>GEOM</ogc:PropertyName>
            <gml:Box>
               <gml:coordinates>1,40 12,56</gml:coordinates>
            </gml:Box>
         </ogc:BBOX>
      </ogc:Filter>
   </Query>
</GetFeature>
```

The Web Authentication Service (WAS) is defined by GDI-NRWA as a service that is able to process GetSession requests and respond with a SessionID that is valid for a defined period of time. deegree extends the functionality of WAS by the operation DescribeUser that allows to identify the user of a SessionID. A detailed description of WAS is available as a separate document.

3.2.2 GeneralConditions

In section <GeneralConditions> HTTP-based rules are defined that are supposed to prevent that requests which can already because of their 'exterior' format identified as meaningless or serious, are relayed to the secured service. The release of specific value ranges for specific HTTP-related parameters corresponds to the form used for service-specific parameters. Therefore, the following description is representative for all service specific parameter definitions describe later on.

Inside the element <GeneralConditions> is the <Conditions> element including a list of four parameter definitions, for which valid value ranges have to be defined. Each parameter has the two attributes 'name' and 'userCoupled'. The first attribute defines the name of the parameter for which a value range will be supplied. Using the attribute 'userCoupled' it is possible to declare if validation of the valid value range shall be done user-specific or not. If userCoupled is set to 'false', the value range defined using <Value> elements is valid for all users. If userCoupled=”true”, U3R is used to determine if the parameters used inside a request are valid.

Alternatively there is a list of <Value> elements embedded inside <Parameter> or the empty element <Any/>. Using <Value>, valid values for a parameter can be determined. If <Any/> is used instead, each value for this parameter is accepted inside a request. If neither <Value> or <Any/> is defined, owsProxy does not accept any value at all for this parameter. This can for example be of use to refuse the right of usage of a specific parameter or function to anonymous users, but allow usage to known users.

Currently four parameters inside of GeneralConditions can be defined:

1. getContentLength: maximum number of allowed characters in one HTTP-Get request
2. postContentLength: maximum number of allowed characters in one HTTP-Post request
3. header: allowed HTTP-Header Parameter (currently not implemented)
4. requestMethod: comma seperated list of allowed HTTP methods (the standard behaviour of OWS is to only support GET and POST)

3.3 WMS Policies

A policy for a WMS filter complements the generic rules and conditions defined in the above chapters by rules that are specific to WMS. The definition of request-specific conditions and rules for modification of responses of a WMS belong in this category. The following example shows the policy for protecting a WMS.

```
<?xml version="1.0" encoding="UTF-8"?>
<dgsec:OWSPolicy service="WMS" xmlns:dgsec="http://www.deegree.org/security"
   xmlns:xlink="http://www.w3.org/1999/xlink">
   <dgsec:Security>
<dgsec:RegistryClass>org.deegree.security.drm.SQLRegistry</dgsec:RegistryClass>
      <dgsec:ReadWriteTimeout>300</dgsec:ReadWriteTimeout>
      <dgsec:RegistryConfig>
         <dgjdbc:JDBCConnection xmlns:dgjdbc="http://www.deegree.org/jdbc">
            <dgjdbc:Driver>oracle.jdbc.OracleDriver</dgjdbc:Driver>
            <dgjdbc:Url>dgjdbc:oracle:thin:@localhost:1521:latlon</dgjdbc:Url>
            <dgjdbc:User>system</dgjdbc:User>
            <dgjdbc:Password>latlondba01</dgjdbc:Password>
            <dgjdbc:SecurityConstraints/>
            <dgjdbc:Encoding>iso-8859-1</dgjdbc:Encoding>
         </dgjdbc:JDBCConnection>
      </dgsec:RegistryConfig>
   </dgsec:Security>
   <dgsec:GeneralConditions>
      <dgsec:Conditions>
         <dgsec:Parameter name="getContentLength" userCoupled="false">
            <dgsec:Value>1000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="postContentLength" userCoupled="false">
            <dgsec:Value>10000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="header" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="requestMethod" userCoupled="false">
            <dgsec:Value>GET,POST</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:Conditions>
   </dgsec:GeneralConditions>
   <dgsec:Requests>
      <dgsec:GetCapabilities>
         <dgsec:PreConditions>
            <dgsec:Parameter name="request" userCoupled="false">
               <dgsec:Value>GetCapabilities</dgsec:Value>
               <dgsec:Value>capabilities</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="version" userCoupled="false">
               <dgsec:Value>1.1.0</dgsec:Value>
               <dgsec:Value>1.1.1</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="updatesequence" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
         </dgsec:PreConditions>
         <dgsec:PostConditions>
            <dgsec:Parameter name="layers" userCoupled="false">
               <dgsec:Value>stadtbezirke</dgsec:Value>
               <dgsec:Value>gebaeude</dgsec:Value>
               <dgsec:Value>quartiere</dgsec:Value>
            </dgsec:Parameter>
         </dgsec:PostConditions>
      </dgsec:GetCapabilities>
      <dgsec:GetMap>
         <dgsec:PreConditions>
            <dgsec:Parameter name="request" userCoupled="false">
               <dgsec:Value>GetMap</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="layers" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
            <dgsec:Parameter name="version" userCoupled="false">
               <dgsec:Value>1.1.0</dgsec:Value>
               <dgsec:Value>1.1.1</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="bgcolor" userCoupled="false">
```

```
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="transparency" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="exception" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="bbox" userCoupled="false">
            <dgsec:Value>3411000,5829000,3690000,6028000,EPSG:31467</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="format" userCoupled="false">
            <dgsec:Value>image/jpeg</dgsec:Value>
            <dgsec:Value>image/png</dgsec:Value>
            <dgsec:Value>image/gif</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="maxWidth" userCoupled="false">
            <dgsec:Value>1600</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="maxHeight" userCoupled="false">
            <dgsec:Value>1600</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="resolution" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:GetMap>
   <dgsec:GetFeatureInfo>
      <dgsec:PreConditions>
         <dgsec:Parameter name="request" userCoupled="false">
            <dgsec:Value>GetFeatureInfo</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="version" userCoupled="false">
            <dgsec:Value>1.1.0</dgsec:Value>
            <dgsec:Value>1.1.1</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="featureInfoLayers" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="infoFormat" userCoupled="false">
            <dgsec:Value>application/vnd.ogc.gml</dgsec:Value>
            <dgsec:Value>text/html</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="maxFeatureCount" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="exception" userCoupled="false">
            <!-- any exception format defined here must also be valid
                 for GetMap -->
            <dgsec:Any/>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:GetFeatureInfo>
   <dgsec:GetLegendGraphic>
      <dgsec:PreConditions>
         <dgsec:Parameter name="request" userCoupled="false">
            <dgsec:Value>GetLegendGraphic</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="version" userCoupled="false">
            <dgsec:Value>1.1.1</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="layers" userCoupled="false">
            <dgsec:Any/>
```

```
            </dgsec:Parameter>
            <dgsec:Parameter name="maxWidth" userCoupled="false">
               <dgsec:Value>50</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="maxHeight" userCoupled="false">
               <dgsec:Value>50</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="format" userCoupled="false">
               <dgsec:Value>image/jpeg</dgsec:Value>
               <dgsec:Value>image/png</dgsec:Value>
               <dgsec:Value>image/gif</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="exception" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
         </dgsec:PreConditions>
         <dgsec:PostConditions>
            <dgsec:Any/>
         </dgsec:PostConditions>
      </dgsec:GetLegendGraphic>
   </dgsec:Requests>
</dgsec:OWSPolicy>
```

Following the elements <Security> and <GeneralConditions> the section <Request> is declared. Here, all requests to be processed by owsProxy are listed. If a request is missing here ins spite of the fact that the protected service would generally be able to process it, each corresponding request will be rejected by owsProxy as not valid. Each registered request encompasses the two sub elements <PreConditions> and <PostConditions>. Inside of these elements the conditions are defined that a incoming request has to fulfil to be accepted. The syntax of these definitions corresponds to the one already described for <GeneralConditions>.

As the conditions defined inside of <PreConditions> relate to incoming request, respectively to the validation before the protected service is addressed. <PostConditions> on the other hand corresponds to post-processing of the responses delivered by the protected service. This can be shown very well at the example of a GetCapabilities-request.

Besides the service type (service=WMS), a GetCapabilities-request to a OGC WMS has to include the parameter REQUEST (e.g. REQUEST=GetCapabilities). The usage of VERSION (e.g. VERSION=1.1.0) and UPDATESEQUENCE (e.g. UPDATESEQUENCE=1234) is optional. In the example shown above <PreConditions> for GetCapabilities are declared so that requests with the REQUEST-parameter 'GetCapabilities' and 'capabilities' as well as the version numbers '1.1.0' and '1.1.1' are accepted. There is no condition defined for UPDATESEQUENCE (</Any>).

In response to a valid GetCapabilities-request a WMS sends a Capabilities-document that among other things includes a list of all available layers. If not all of these layers shall be returned to a requesting client, this can be declared using the parameter 'layers'. This parameter can include a positive list of all layers, that are passed on by owsProxy. All other layers are filtered out of the Capabilities-document. So it is for example possible to create thematic views on a WMS or to define that the visible content is dependant on the net segment out of which the request is send. Also, the list of visible layers can be coupled to individual user rights if.

```
<parameter name="layers" userCoupled="true"/>
```

is defined.

If neither <PreConditions> nor <PostConditions> should be restricted in any way, the parameter <Any/> can be used.

In the following the parameters are described for which conditions can currently be configured. If not specified further, their meaning is defined in the OGC WMS specification.

3.3.1 Parameters for GetCapabilities:

preConditions-Parameter:

- request: has to be 'GetCapabilities' or <Any/>
- version: allowed values for requested version numbers
- updatesequence: allowed value for requested update sequence

postConditions-Parameter

- layers (list of valid WMS-Layer): list of those layers, that are passed on by owsProxy

3.3.2 Parameters for GetMap

preConditions-Parameter:

- request: currently has to be 'GetMap' or <Any/>
- version: allowed value for the requested version number
- layers; definition of valid layer for a GetMap request. This includes the names of the valid layers as well as the allowed styles. The styles that are valid for a layer are separated by commas and attached to the layername after a '|'. The key any declares that every style is valid. Independent of this, the defaul style is generally valid.

Example.:

```
stadtbezirke|blaugefuellt,rotumrandet

gebaeude|$any$
```

For the layer 'stadtbezirke' the styles blaugefuellt, rotumrandet as well as the default style are allowed. For the layer 'gebaeude' every style is allowed.

- bgcolor: list of allowed background colors
- transparency: list of allowed transparency values
- exception: List if allowed exception formats
- bbox: the requests BoundingBox has to be covered completely in the bbox declared here
- format: list of allowed formats
- maxWidth: maximum width of the map in pixels
- maxHeight: maximum allowed height of the map in pixels
- resolution: best possible resolution of the map measured as the diagonal length of a pixel of the displayed area in meters
- sld: list of allowed URLs used for referencing SLDs
- sld_body (currently only <Any/> is possible here)

postConditions-Parameter

- currently none (<Any/> has to be declared)

3.3.3 Parameters for GetFeatureInfo

As a GetFeatureInfo request includes a copy of the before sent GetMap request, that is also checked for its validity, the conditions set for GetMap are here also valid.

preConditions-Parameter:

- request: has to be 'GetFeatureInfo' or <Any/>
- version: allowed values for requested version number
- featureInfoLayers: List of layers that can be queried using GetFeatureInfo requests
- infoFormat: List of allowed queryable info formats
- maxFeatureCount: maximum number of returned FeatureInfos
- exception: list of all allowed queryable exception formats

postConditions-Parameter

- currently none (<Any/> has to be declared)

3.3.4 Parameter for GetLegendGraphic

preConditions-Parameter:

- request: has to be 'GetLegendGraphic' or <Any/>
- version: admissible values for requested version number
- layers: list of layers that may be queried using GetLegendGraphic requests
- maxWidth:maximum admissible width (in pixel) of legend symbol
- maxHeight: maximum admissible height (in pixel) of legend symbol
- format: list of admissible, formats to be requested
- exception: list of admissible, exception formats to be requested
- sld: list of admissible URLs that can be used to reference SLDs
- currently none (<Any/> has to be declared)

postConditions-Parameter

- currently none (<Any/> has to be declared)

3.4 WFS Policies

The policy file for securing a WFS is in regard to its construction identical to a WMS policy. Merely the possible request and parameter definitions are different. The following example shows the WFS-specific part of a corresponding policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<dgsec:OWSPolicy service="WFS" xmlns:dgsec="http://www.deegree.org/security"
 xmlns:xlink="http://www.w3.org/1999/xlink">
   <dgsec:Security>
<dgsec:RegistryClass>org.deegree.security.drm.SQLRegistry</dgsec:RegistryClass>
      <dgsec:ReadWriteTimeout>300</dgsec:ReadWriteTimeout>
      <dgsec:RegistryConfig>
         <dgjdbc:JDBCConnection xmlns:dgjdbc="http://www.deegree.org/jdbc">
            <dgjdbc:Driver>oracle.jdbc.OracleDriver</dgjdbc:Driver>
            <dgjdbc:Url>dgjdbc:oracle:thin:@localhost:1521:latlon</dgjdbc:Url>
            <dgjdbc:User>system</dgjdbc:User>
            <dgjdbc:Password>latlondba01</dgjdbc:Password>
            <dgjdbc:SecurityConstraints/>
            <dgjdbc:Encoding>iso-8859-1</dgjdbc:Encoding>
         </dgjdbc:JDBCConnection>
      </dgsec:RegistryConfig>
   </dgsec:Security>
   <dgsec:GeneralConditions>
      <dgsec:Conditions>
         <dgsec:Parameter name="getContentLength" userCoupled="false">
            <dgsec:Value>1000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="postContentLength" userCoupled="false">
```

```
            <dgsec:Value>10000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="header" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="requestMethod" userCoupled="false">
            <dgsec:Value>GET,POST</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:Conditions>
   </dgsec:GeneralConditions>
   <dgsec:Requests>
      <dgsec:GetCapabilities>
         <dgsec:PreConditions>
            <dgsec:Parameter name="request" userCoupled="false">
               <dgsec:Value>GetCapabilities</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="version" userCoupled="false">
               <dgsec:Value>1.1.0</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="updatesequence" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
         </dgsec:PreConditions>
         <dgsec:PostConditions>
            <dgsec:Parameter name="featureTypes" userCoupled="false">
               <dgsec:Value>{http://www.deegree.org/app}:FT1</dgsec:Value>
               <dgsec:Value>{http://www.deegree.org/app}:FT2</dgsec:Value>
            </dgsec:Parameter>
         </dgsec:PostConditions>
      </dgsec:GetCapabilities>
      <dgsec:GetFeature>
         <dgsec:PreConditions>
            <dgsec:Parameter name="request" userCoupled="false">
               <dgsec:Value>GetFeature</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="version" userCoupled="false">
               <dgsec:Value>1.1.0</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="featureTypes" userCoupled="false">
               <dgsec:Value>{http://www.deegree.org/app}:FT1</dgsec:Value>
               <dgsec:Value>{http://www.deegree.org/app}:FT2</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="format" userCoupled="false">
               <dgsec:Value>text/xml; subtype=gml/3.1.1</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="maxFeatures" userCoupled="false">

            <dgsec:Any/>
            </dgsec:Parameter>
         </dgsec:PreConditions>
         <deegreeSec:PostConditions>
            <deegreeSec:Parameter name="instanceFilter" userCoupled="true">
               <deegreeSec:ComplexValue>
                  <wfs:Query typeName="app:WPVS"
                     xmlns:app="http://www.deegree.org/app"
                     xmlns:wfs="http://www.opengis.net/wfs"
                     xmlns:ogc="http://www.opengis.net/ogc">
                     <ogc:Filter>
                        <ogc:Or>
                           <ogc:PropertyIsEqualTo>
                              <ogc:PropertyName>app:texturetype</ogc:PropertyName>
                              <ogc:Literal>specific</ogc:Literal>
                           </ogc:PropertyIsEqualTo>
                           <ogc:PropertyIsGreaterThanOrEqualTo>
                              <ogc:PropertyName>app:diffusecolor</ogc:PropertyName>
                              <ogc:Literal>0 0 1</ogc:Literal>
                           </ogc:PropertyIsGreaterThanOrEqualTo>
                        </ogc:Or>
                     </ogc:Filter>
```

```
                </wfs:Query>
             </deegreeSec:ComplexValue>
             <deegreeSec:ComplexValue>
                <wfs:Query typeName="citygml:Building"
                   xmlns:citygml="http://www.citygml.org/citygml/1/0/0"
                   xmlns:gml="http://www.opengis.net/gml"
                   xmlns:wfs="http://www.opengis.net/wfs"
                   xmlns:ogc="http://www.opengis.net/ogc">
                   <ogc:Filter>
                      <ogc:PropertyIsEqualTo>
                         <ogc:PropertyName>gml:name</ogc:PropertyName>
                         <ogc:Literal>Wirtschaftsgebäude</ogc:Literal>
                      </ogc:PropertyIsEqualTo>
                   </ogc:Filter>
                </wfs:Query>
             </deegreeSec:ComplexValue>
          </deegreeSec:Parameter>
       </deegreeSec:PostConditions>
    </dgsec:GetFeature>
    <dgsec:DescribeFeatureType>
       <dgsec:PreConditions>
          <dgsec:Parameter name="request" userCoupled="false">
             <dgsec:Value>DescribeFeatureType</dgsec:Value>
          </dgsec:Parameter>
          <dgsec:Parameter name="version" userCoupled="false">
             <dgsec:Value>1.1.0</dgsec:Value>
          </dgsec:Parameter>
          <dgsec:Parameter name="format" userCoupled="false">
             <dgsec:Value>text/xml; subtype=gml/3.1.1</dgsec:Value>
          </dgsec:Parameter>
          <dgsec:Parameter name="featureTypes" userCoupled="true"/>
       </dgsec:PreConditions>
       <dgsec:PostConditions>
          <dgsec:Any/>
       </dgsec:PostConditions>
    </dgsec:DescribeFeatureType>
 </dgsec:Requests>
</dgsec:OWSPolicy>
```

3.4.1 Parameter for GetCapabilities:

preConditions-Parameter:

- request: currently has to be 'GetCapabilities' or <Any/>
- version: admissible values for the requested version number
- updatesequence: admissible values for the requested Updatesequence

postConditions-Parameter

- featureTypes: List of valid, usually qualified FeatureType names ({namespace}:Name)

3.4.2 Parameters for GetFeature

At this time there is no possibility to check a filter that might be connected with a GetFeature request for its validity. In a future version of owsProxy it will be possible to define OGC-Filter constructs for chosen FeatureTypes. These will then be combined with a incoming request using AND therefore restricting the amount of accessible features in general or user-specific.

preConditions-Parameter:

- request: currently has to be 'GetFeature' or <Any/>
- version: admissible values for the requested version number
- featureTypes: list of valid, usually qualified FeatureType names ({namespace}:Name)
- format: admissible query formats
- maxFeatures

postConditions-Parameter

- instanceFilter: OGC WFS queries defining which feature instances may pass owsProxy (see above example)
- horizontalAccuracy: maximum admissible horizontal accuracy
- verticalAccuracy: maximum admissible vertical accuracy.
- XSLT: reference to a XSLT script that can be used to apply complex filters

3.4.3 Parameter for DescribeFeatureType

preConditions-Parameter:

- request: currently has to be 'DescribeFeatureType' or <Any/>
- version: admissible values for queried version numbers
- featureTypes (corresponds to typeName of the WFS Spezifikation)

postConditions-Parameter

- currentyl none (<Any/> has to be set)

3.4.4 Parameter for WFS_Insert

PreConditions-Parameter:

- typeName: list of valid, usually qualified FeatureType names ({namespace}:Name)

postConditions-Parameter

- currentyl none (<Any/> has to be set)

3.4.5 Parameter for WFS_Update

PreConditions-Parameter:

- typeName: list of valid, usually qualified FeatureType names ({namespace}:Name)

postConditions-Parameter

- instanceFilter: OGC WFS queries defining which feature instances may pass owsProxy (see above example)

3.4.6 Parameter für WFS_Delete

PreConditions-Parameter:

- typeName: list of valid, usually qualified FeatureType names ({namespace}:Name)

postConditions-Parameter

- instanceFilter: OGC WFS queries defining which feature instances may pass owsProxy (see above example)

3.4.7 Usercoupled postConditions

If postConditions for GetFeature, WFS_Delete or WFS_Update are defined as userCoupled, the queries declared using the parameter instanceFilter have to be included in an encapsulating XML expression.

```
<ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">
   <ogc:And>
      <ogc:PropertyIsEqualTo>
         <ogc:PropertyName>horizontalAccuracy</ogc:PropertyName>
         <ogc:Literal>10</ogc:Literal>
      </ogc:PropertyIsEqualTo>
      <ogc:PropertyIsEqualTo>
         <ogc:PropertyName>verticalAccuracy</ogc:PropertyName>
         <ogc:Literal>1</ogc:Literal>
      </ogc:PropertyIsEqualTo>
      <ogc:PropertyIsEqualTo>
         <ogc:PropertyName>xslt</ogc:PropertyName>
         <ogc:Literal>file:/d:/temp/test.xsl</ogc:Literal>
      </ogc:PropertyIsEqualTo>
      <ogc:PropertyIsEqualTo>
         <ogc:PropertyName>instanceFilter</ogc:PropertyName>
         <ogc:Literal>
             <![CDATA[<ogc:Filter xmlns:ogc="http://www.opengis.net/ogc"
                 xmlns:app="http://www.deegree.org/app">
         <ogc:And>
            <ogc:PropertyIsGreaterThan>
               <ogc:PropertyName>app:dateOfBirth</ogc:PropertyName>
               <ogc:Literal>1820</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
            <ogc:PropertyIsEqualTo>
                <ogc:PropertyName>
                   app:placeOfBirth/app:Place/app:country/app:Country/app:name
                 </ogc:PropertyName>
               <ogc:Literal>Germany</ogc:Literal>
            </ogc:PropertyIsEqualTo>
         </ogc:And>
      </ogc:Filter>]]></ogc:Literal>
      </ogc:PropertyIsEqualTo>
   </ogc:And>
</ogc:Filter>
```

As can be seen her, the literal of the last PropertyIsEqualTo expression is embedded in a CDATA section. At this place, the CDATA section is mandatory as a literal may only include a string. The interpretation of the filter expression is a task of the owsProxy.

3.5 CSW Policies

A policy file for securing a CSW is in regard to its construction identical to the other policy files. Only the possible request and parameter definitions are different. The following example shows the CSW specific excerpt of a corresponding policy.

```
<?xml version="1.0" encoding="UTF-8"?>
<dgsec:OWSPolicy xmlns:dgsec="http://www.deegree.org/security" service="CSW">
   <dgsec:Security>
<dgsec:RegistryClass>org.deegree.security.drm.SQLRegistry</dgsec:RegistryClass>
      <dgsec:ReadWriteTimeout>300</dgsec:ReadWriteTimeout>
      <dgsec:RegistryConfig>
         <dgjdbc:JDBCConnection xmlns:dgjdbc="http://www.deegree.org/jdbc">
            <dgjdbc:Driver>oracle.jdbc.OracleDriver</dgjdbc:Driver>
            <dgjdbc:Url>dgjdbc:oracle:thin:@localhost:1521:latlon</dgjdbc:Url>
            <dgjdbc:User>system</dgjdbc:User>
            <dgjdbc:Password>latlondba01</dgjdbc:Password>
            <dgjdbc:SecurityConstraints/>
            <dgjdbc:Encoding>iso-8859-1</dgjdbc:Encoding>
         </dgjdbc:JDBCConnection>
      </dgsec:RegistryConfig>
   </dgsec:Security>
   <dgsec:GeneralConditions>
      <dgsec:Conditions>
         <dgsec:Parameter name="getContentLength" userCoupled="false">
            <dgsec:Value>1000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="postContentLength" userCoupled="false">
            <dgsec:Value>1000000</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter name="header" userCoupled="false">
            <dgsec:Any/>
         </dgsec:Parameter>
         <dgsec:Parameter name="requestMethod" userCoupled="false">
            <dgsec:Value>GET,POST</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:Conditions>
   </dgsec:GeneralConditions>
   <dgsec:Requests>
      <dgsec:GetCapabilities>
         <dgsec:PreConditions>
            <dgsec:Parameter name="request" userCoupled="false">
               <dgsec:Value>GetCapabilities</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="version" userCoupled="false">
               <dgsec:Value>2.0.0</dgsec:Value>
            </dgsec:Parameter>
            <dgsec:Parameter name="updatesequence" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
            <dgsec:Parameter name="sections" userCoupled="false">
               <dgsec:Any/>
            </dgsec:Parameter>
         </dgsec:PreConditions>
         <dgsec:PostConditions>
            <dgsec:Any/>
         </dgsec:PostConditions>
      </dgsec:GetCapabilities>
```

```
<dgsec:DescribeRecord>
   <dgsec:PreConditions>
      <dgsec:Parameter userCoupled="false" name="typeName">
         <dgsec:Value>csw:record</dgsec:Value>
         <dgsec:Value>csw:profile</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter userCoupled="false" name="schemaLanguage">
         <dgsec:Any/>
      </dgsec:Parameter>
      <dgsec:Parameter userCoupled="false" name="outputFormat">
         <dgsec:Value>text/xml</dgsec:Value>
      </dgsec:Parameter>
   </dgsec:PreConditions>
   <dgsec:PostConditions>
      <dgsec:Any/>
   </dgsec:PostConditions>
</dgsec:DescribeRecord>
<dgsec:GetRecords>
   <dgsec:PreConditions>
      <dgsec:Parameter name="version" userCoupled="false">
         <dgsec:Value>2.0.0</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="maxRecords" userCoupled="false">
         <dgsec:Value>1000</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="metadataFormat" userCoupled="false">
         <dgsec:Any/>
      </dgsec:Parameter>
      <dgsec:Parameter name="outputFormat" userCoupled="false">
         <dgsec:Any/>
      </dgsec:Parameter>
      <dgsec:Parameter name="outputSchema" userCoupled="false">
         <dgsec:Any/>
      </dgsec:Parameter>
      <dgsec:Parameter name="resultType" userCoupled="false">
         <dgsec:Value>HITS</dgsec:Value>
         <dgsec:Value>hits</dgsec:Value>
         <dgsec:Value>RESULTS</dgsec:Value>
         <dgsec:Value>results</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="sortBy" userCoupled="false">
         <dgsec:Value>Title</dgsec:Value>
         <dgsec:Value>Date</dgsec:Value>
         <dgsec:Value>Envelope</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="typeNames" userCoupled="false">
         <dgsec:Value>csw:dataset</dgsec:Value>
         <dgsec:Value>csw:datasetcollection</dgsec:Value>
         <dgsec:Value>csw:service</dgsec:Value>
         <dgsec:Value>csw:application</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="elementSetName" userCoupled="false">
         <dgsec:Value>brief</dgsec:Value>
         <dgsec:Value>summary</dgsec:Value>
         <dgsec:Value>full</dgsec:Value>
      </dgsec:Parameter>
   </dgsec:PreConditions>
   <dgsec:PostConditions>
      <dgsec:Any/>
   </dgsec:PostConditions>
</dgsec:GetRecords>
<dgsec:GetRecordById>
   <dgsec:PreConditions>
      <dgsec:Parameter name="version" userCoupled="false">
         <dgsec:Value>2.0.0</dgsec:Value>
      </dgsec:Parameter>
      <dgsec:Parameter name="elementSetName" userCoupled="false">
         <dgsec:Value>brief</dgsec:Value>
         <dgsec:Value>summary</dgsec:Value>
```

```
            <dgsec:Value>full</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:GetRecordById>
   <dgsec:CSW_Insert>
      <dgsec:PreConditions>
         <dgsec:Parameter userCoupled="false" name="metadataFormat">

<dgsec:Value>{http://metadata.dgiwg.org/smXML}:MD_Metadata</dgsec:Value>

<dgsec:Value>{http://schemas.opengis.net/iso19115full}:MD_Metadata</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:CSW_Insert>
   <dgsec:CSW_Update>
      <dgsec:PreConditions>
         <dgsec:Parameter userCoupled="false" name="metadataFormat">
   <dgsec:Value>{http://metadata.dgiwg.org/smXML}:MD_Metadata</dgsec:Value>
<dgsec:Value>{http://schemas.opengis.net/iso19115full}:MD_Metadata</dgsec:Value>
         </dgsec:Parameter>
         <dgsec:Parameter userCoupled="false" name="typeName">
            <dgsec:Value>csw:dataset</dgsec:Value>
            <dgsec:Value>csw:datasetcollection</dgsec:Value>
            <dgsec:Value>csw:series</dgsec:Value>
            <dgsec:Value>csw:application</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:CSW_Update>
   <dgsec:CSW_Delete>
      <dgsec:PreConditions>
         <dgsec:Parameter userCoupled="false" name="typeName">
            <dgsec:Value>csw:dataset</dgsec:Value>
            <dgsec:Value>csw:datasetcollection</dgsec:Value>
            <dgsec:Value>csw:series</dgsec:Value>
            <dgsec:Value>csw:application</dgsec:Value>
         </dgsec:Parameter>
      </dgsec:PreConditions>
      <dgsec:PostConditions>
         <dgsec:Any/>
      </dgsec:PostConditions>
   </dgsec:CSW_Delete>
</dgsec:Requests>
</dgsec:OWSPolicy>
```

The securing of the operations GetCapabilities, GetRecords, DescribeRecord und GetRecordById is suppored. Additionally the securing of transactional requests is possible. As these can be grouped in Insert-, Update- and Delete operations, separate value ranges for the separate parameters are defined.

3.5.1 Parameter for GetCapabilities

preConditions-Parameter:

- request
- version
- updatesequence

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.2 Parameter for DescribeRecord (currently only partially supported)

preConditions-Parameter:

- version
- typeName
- schemaLanguage
- outputFormat

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.3 Parameter for GetRecords

Currently it is not possible to check the filter expression that might be associated with a GetRecords request for its validity. In a future version of owsProxy it will be possible to define OGC-Filter constructs for chosen FeatureTypes. These will then be combined with a incoming request using AND therefore restricting the amount of accessible features in general or user-specific.

preConditions-Parameter:

- request: currently has to be 'GetRecords' or <Any/>
- version: admissible values for requested version numbers
- maxRecords: maximum admissible number of requested records
- metadataFormat: list of admissible requestable metadata formats
- outputFormat: list of admissible requestable formats (e.g: text/xml)
- outputSchema:
- resultType: list of requestable ResultTypes (HITS | RESULTS)
- sortBy: list of properties (fully qualified names) that can be used for sorting results
- typeNames: list of queryable typeName (e.g. csw:record)

- elementSetName: list of queryable elementSetNames (e.g. full)

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.4 Parameters for GetRecordById

preConditions-Parameter:

- version: admissible values for requestes version numbers
- elementSetName: list of admissible elementSetNames (e.g. full)

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.5 Parameter for CSW_Insert

preConditions-Parameter:

- metadataFormat

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.6 Parameter for CSW_Update

preConditions-Parameter:

- metadataFormat: list of admissible queryable metadata formats
- typeName: list of queryble typeNemes (e.g.. csw:record)

postConditions-Parameter

- currently none (<Any/> has to be set)

3.5.7 Parameter for CSW_Delete

preConditions-Parameter:

- typeName: list of queryable typeNames (e.g. csw:record)

postConditions-Parameter

- currently none (<Any/> has to be set)

4 Appendix (XML-Schema Files)

4.1 General

```
<xs:schema targetNamespace="http://www.deegree.org/jdbc"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:jdbc="http://www.deegree.org/jdbc" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:complexType name="JDBCConnectionType">
      <xs:annotation>
         <xs:documentation>
       For JDBC based Datastores only, contains the necessary connection
information.
       </xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="Driver" type="xs:string"/>
         <xs:element name="Url" type="xs:string"/>
         <xs:element name="User" type="xs:string" minOccurs="0"/>
         <xs:element name="Password" type="xs:string" minOccurs="0"/>
         <xs:element name="SecurityConstraints" minOccurs="0"/>
         <xs:element name="Encoding" minOccurs="0"/>
         <xs:element name="AliasPrefix" minOccurs="0"/>
         <xs:element name="SDEDatabase" minOccurs="0">
            <xs:annotation>
               <xs:documentation>
                        Only applies to ArcSDE connections.
                     </xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="SDEVersion" minOccurs="0">
            <xs:annotation>
               <xs:documentation>
                        Only applies to ArcSDE connections.
                     </xs:documentation>
            </xs:annotation>
         </xs:element>
         <xs:element name="OracleWorkspace" minOccurs="0">
            <xs:annotation>
               <xs:documentation>
          Name of the workspace to switch to on connect. Only applies to Oracle
10g. Default workspace is "LIVE".
          </xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
   <xs:element name="JDBCConnection" type="jdbc:JDBCConnectionType"/>
</xs:schema>

<xs:schema targetNamespace="http://www.deegree.org/security"
xmlns:jdbc="http://www.deegree.org/jdbc"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:deegree="http://www.deegree.org/security"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- import OGC commons to use some basic types-->
   <xs:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="./xlinks.xsd"/>
   <xs:import namespace="http://www.deegree.org/jdbc"
schemaLocation="./jdbc_connection.xsd"/>
   <xs:element name="Policy" type="deegree:PolicyType">
      <xs:annotation>
         <xs:documentation>root element of the deegree policy document for
protecting Web map Services</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="PolicyType">
```

```
      <xs:sequence>
         <xs:element name="Security" type="deegree:SecurityType" minOccurs="0"/>
         <xs:element name="GeneralConditions" type="deegree:GeneralConditionsType"
minOccurs="0"/>
         <xs:element name="Requests" type="deegree:RequestsType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="service" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:complexType name="RequestsType">
      <xs:sequence>
         <xs:element ref="deegree:_Request" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="SecurityType">
      <xs:annotation>
         <xs:documentation>cofiguration off the access to the security management
system/database</xs:documentation>
      </xs:annotation>
      <xs:sequence>
         <xs:element name="RegistryClass" type="xs:string"/>
         <xs:element name="ReadWriteTimeout" type="xs:int" default="500"/>
         <xs:element name="RegistryConfig" type="deegree:RegistryConfigType"/>
         <xs:element name="AuthentificationSettings"
type="deegree:AuthentificationSettingsType" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="RegistryConfigType">
      <xs:sequence>
         <xs:element ref="jdbc:JDBCConnection" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="GeneralConditionsType">
      <xs:choice minOccurs="0">
         <xs:element name="Conditions" type="deegree:ConditionType" minOccurs="0"/>
         <xs:element name="any" type="xs:string" minOccurs="0">
            <xs:annotation>
               <xs:documentation>the presents of this element indicates that no
constraints are made</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:complexType name="_RequestType" abstract="true">
      <xs:annotation>
         <xs:documentation>A concrete _requestType will encapsulate and describe the
conditions for
         a request and the repsonse to it for a OWS.  Also a concrete _requestType
must be named like
         the request which conditions will be described</xs:documentation>
      </xs:annotation>
      <xs:choice minOccurs="0">
         <xs:sequence>
            <xs:element name="PreConditions" type="deegree:ConditionType"
minOccurs="0"/>
            <xs:element name="PostConditions" type="deegree:ConditionType"
minOccurs="0"/>
         </xs:sequence>
         <xs:element name="Any" type="xs:string" minOccurs="0">
            <xs:annotation>
               <xs:documentation>the presents of this element indicates that no
constraints are made</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:element name="_Request" type="deegree:_RequestType" abstract="true"/>
   <xs:complexType name="AuthentificationSettingsType">
      <xs:sequence>
```

```
         <xs:element name="AuthentificationService"
type="deegree:AuthentificationServiceType"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="AuthentificationServiceType">
      <xs:sequence>
         <xs:element name="OnlineResource" type="deegree:OnlineResourceType"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="OnlineResourceType">
      <xs:attributeGroup ref="xlink:simpleLink"/>
   </xs:complexType>
   <xs:complexType name="GetCapabilitiesType">
      <xs:annotation>
         <xs:documentation>GetCapabilties will be used by all OWS so it defined in
the base schema</xs:documentation>
      </xs:annotation>
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetCapabilities" type="deegree:GetCapabilitiesType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="ConditionType">
      <xs:annotation>
         <xs:documentation>If nor parameter or any is defined no request will
fullfill a condintion. This realizes
         the concept that a security filter will block everything except it
especially allowed</xs:documentation>
      </xs:annotation>
      <xs:choice minOccurs="0">
         <xs:element ref="deegree:_Parameter" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="Any" type="xs:string" minOccurs="0">
            <xs:annotation>
               <xs:documentation>the presents of this element indicates that no
constraints are made</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:choice>
   </xs:complexType>
   <xs:element name="_Condition" type="deegree:ConditionType" abstract="true"/>
   <xs:complexType name="ParameterType">
      <xs:choice>
         <xs:sequence>
            <xs:element name="Value" type="xs:string" minOccurs="0"
maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>Unordered list of all the valid values for this
parameter or other quantity.
                     For those parameters that contain a list or sequence of
values, these values shall be for
                     individual values in the list. The allowed set of values and
the allowed server restrictions
                     on that set of values shall be specified in the
Implementation Specification for this service. </xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:element name="ComplexValue" minOccurs="0" maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>
                     a condition may contains complex values like filter expressions
that can be put
                     into a ComplexValue element as part of a condition
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType name="ExtensionType">
                  <xs:sequence>
                     <xs:any namespace="##any"/>
                  </xs:sequence>
```

```
            </xs:complexType>
         </xs:element>
      </xs:sequence>
      <xs:element name="Any" type="xs:string" minOccurs="0">
         <xs:annotation>
            <xs:documentation>the presents of this element indicates that no
constraints are made on a parameter</xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:choice>
   <xs:attribute name="userCoupled" type="xs:boolean" default="false"/>
</xs:complexType>
<xs:element name="_Parameter" type="deegree:ParameterType" abstract="true"/>
</xs:schema>
```

4.2 WMS Policy

```
<xs:schema targetNamespace="http://www.deegree.org/security"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:deegree="http://www.deegree.org/security" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:include schemaLocation="deegreePolicy.xsd"/>
   <xs:element name="OWSPolicy" type="deegree:PolicyType">
      <xs:annotation>
         <xs:documentation>root element of the deegree policy document for
protecting Web map Services</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="GetMapType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetMap" type="deegree:GetMapType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="GetFeatureInfoType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetFeatureInfo" type="deegree:GetFeatureInfoType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="GetLegendGraphicType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetLegendGraphic" type="deegree:GetLegendGraphicType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="GetScaleBarType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetScaleBar" type="deegree:GetScaleBarType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="WMSParameterType">
      <xs:complexContent>
         <xs:extension base="deegree:ParameterType">
            <xs:attribute name="name" type="deegree:WmsParamNameType"
use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="Parameter" type="deegree:WMSParameterType"
substitutionGroup="deegree:_Parameter"/>
   <xs:simpleType name="WmsParamNameType">
      <xs:restriction base="xs:string">
```

```
            <xs:enumeration value="requestMethod"/>
            <xs:enumeration value="getContentLength"/>
            <xs:enumeration value="postContentLength"/>
            <xs:enumeration value="header"/>
            <xs:enumeration value="request"/>
            <xs:enumeration value="version"/>
            <xs:enumeration value="updatesequence"/>
            <xs:enumeration value="layers"/>
            <xs:enumeration value="bbox"/>
            <xs:enumeration value="resolution"/>
            <xs:enumeration value="maxWidth"/>
            <xs:enumeration value="maxHeight"/>
            <xs:enumeration value="transparency"/>
            <xs:enumeration value="bgcolor"/>
            <xs:enumeration value="color"/>
            <xs:enumeration value="labelColor"/>
            <xs:enumeration value="bottomLabel"/>
            <xs:enumeration value="topLabel"/>
            <xs:enumeration value="maxSize"/>
            <xs:enumeration value="sld"/>
            <xs:enumeration value="sld_body"/>
            <xs:enumeration value="format"/>
            <xs:enumeration value="exception"/>
            <xs:enumeration value="featureInfoLayers"/>
            <xs:enumeration value="infoFormat"/>
            <xs:enumeration value="maxFeatureCount"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:schema>
```

4.3 WFS Policy

```
<xs:schema targetNamespace="http://www.deegree.org/security"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:deegree="http://www.deegree.org/security" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:include schemaLocation="deegreePolicy.xsd"/>
   <xs:element name="OWSPolicy" type="deegree:PolicyType">
      <xs:annotation>
         <xs:documentation>root element of the deegree policy document for
protecting Web map Services</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="GetFeatureType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetFeature" type="deegree:GetFeatureType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="DescribeFeatureTypeType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="DescribeFeatureType" type="deegree:DescribeFeatureTypeType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="WFSParameterType">
      <xs:complexContent>
         <xs:extension base="deegree:ParameterType">
            <xs:attribute name="name" type="deegree:WFSParamNameType"
use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:complexType name="InsertType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
```

```
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="WFS_Insert" type="deegree:InsertType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="UpdateType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="WFS_Update" type="deegree:UpdateType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="DeleteType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="WFS_Delete" type="deegree:DeleteType"
substitutionGroup="deegree:_Request"/>
   <xs:element name="Parameter" type="deegree:WFSParameterType"
substitutionGroup="deegree:_Parameter"/>
   <xs:simpleType name="WFSParamNameType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="requestMethod"/>
         <xs:enumeration value="getContentLength"/>
         <xs:enumeration value="postContentLength"/>
         <xs:enumeration value="header"/>
         <xs:enumeration value="request"/>
         <xs:enumeration value="version"/>
         <xs:enumeration value="updatesequence"/>
         <xs:enumeration value="featureTypes"/>
         <xs:enumeration value="bbox"/>
         <xs:enumeration value="format"/>
         <xs:enumeration value="exception"/>
         <xs:enumeration value="maxFeatures"/>
         <xs:enumeration value="typeName"/>
         <xs:enumeration value="resultType"/>
         <xs:enumeration value="traverseXLinkDepth"/>
         <xs:enumeration value="traverseXLinkExpiry"/>
         <xs:enumeration value="instanceFilter"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

4.4 CSW Policy

```
<xs:schema targetNamespace="http://www.deegree.org/security"
xmlns:deegree="http://www.deegree.org/security"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:include schemaLocation="deegreePolicy.xsd"/>
   <xs:element name="OWSPolicy" type="deegree:PolicyType">
      <xs:annotation>
         <xs:documentation>root element of the deegree policy document for
protecting Catalogue Service Web Profile</xs:documentation>
      </xs:annotation>
   </xs:element>
   <xs:complexType name="GetRecordsType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="GetRecords" type="deegree:GetRecordsType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="GetRecordByIdType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
```

```
   <xs:element name="GetRecordById" type="deegree:GetRecordByIdType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="DescribeRecordType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="DescribeRecord" type="deegree:DescribeRecordType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="InsertType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="CSW_Insert" type="deegree:InsertType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="UpdateType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="CSW_Update" type="deegree:UpdateType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="DeleteType">
      <xs:complexContent>
         <xs:extension base="deegree:_RequestType"/>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="CSW_Delete" type="deegree:DeleteType"
substitutionGroup="deegree:_Request"/>
   <xs:complexType name="CSWParameterType">
      <xs:complexContent>
         <xs:extension base="deegree:ParameterType">
            <xs:attribute name="name" type="deegree:CSWParamNameType"
use="required"/>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:element name="Parameter" type="deegree:CSWParameterType"
substitutionGroup="deegree:_Parameter"/>
   <xs:simpleType name="CSWParamNameType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="requestMethod"/>
         <xs:enumeration value="getContentLength"/>
         <xs:enumeration value="postContentLength"/>
         <xs:enumeration value="header"/>
         <xs:enumeration value="request"/>
         <xs:enumeration value="version"/>
         <xs:enumeration value="updatesequence"/>
         <xs:enumeration value="resultType"/>
         <xs:enumeration value="maxRecords"/>
         <xs:enumeration value="namespace"/>
         <xs:enumeration value="outputFormat"/>
         <xs:enumeration value="outputSchema"/>
         <xs:enumeration value="metadataFormat"/>
         <xs:enumeration value="typeNames"/>
         <xs:enumeration value="elementSetName"/>
         <xs:enumeration value="sortBy"/>
         <xs:enumeration value="sections"/>
         <xs:enumeration value="distributedSearch"/>
         <xs:enumeration value="responseHandler"/>
         <xs:enumeration value="typeName"/>
         <xs:enumeration value="schemaLanguage"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```