deegree Web Coverage Service 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

Date	Description	Author
2007-01-09	Update using new formatting style	Markus Müller
2007-06-08	Update on demo release candidate v2.1 rc1	Hanko Rubach
2008-04-15	Update to v2.2	Judit Mays
2009-02-06	prepared for deegree 2.3 version / TODOs added	Andreas Poth
2009-12-02	Updates for v2.3; mainly: hint on demo pages, add example for Oracle Georaster (chapter 4.3.3)	Sebastian Goerke
2010-01-14	Description for dynamic datasources added	Andreas Poth
2010-01-20	Update path for CoverageDescription	Sebastian Goerke

Table of Contents

Index of Tables

Illustration Index

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.

deegree2.3 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 Coverage Service (WCS), an implementation of OGC's Web Coverage Service Implementation Specification 1.0.0 (without corrigendum). deegree WCS is the official Reference Implementation of the OGC for the mentioned standard.

deegree's Web Coverage Service (WCS) is able to read coverages from different storage formats and deliver it to any client that is able to perform an according HTTP GET or POST request. At the moment supported formats are limited to several raster data formats; but in general a coverage has **not** to be a raster dataset at all.

Besides the WCS, 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 Coverage Server is very flexible concerning the possibilities of configuration, the adaption to different data sources and formats, layouts and server environments. The configuration of WCS is similar to configuration of the other deegree Web Services and requires customizing different XML files that control the functionality of the deegree server.

The web services of deegree are realized as Java modules controlled by one central servlet (a "dispatcher"). This servlet has to be integrated into the respective application server/servlet engine. Most of the common application servers support servlet technology, so deegree is not limited to special products. The Apache-Tomcat 5.5.x 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 Feature Service to run 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.

2.2 deegree Web Coverage Service Release

deegree Web Coverage 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 WCS 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 (see also 6.1.1 Manual Tomcat configuration). Because of this possibility, in the remainder of this document, the directory you extracted the files to is referred to as `$wcs_home$` (=`$TOMCAT_HOME$/webapps/deegree-wcs` in the standard case).

Your `$wcs_home$` will contain the following structure:

directory	Content
./WEB-INF	Required by Tomcat, containing all libraries, configuration- and data-files
./WEB-INF/conf/wcs	WCS configuration files
./WEB-INF/scripts	RasterTreeBuilder
./WEB-INF/data	Example Data

Table 1: Directory structure of the WCS release

2.3 Testing the installation

deegree WCS comes with one test coverages (saltlakecity) covering the city with a digital orthogonal photo.

In the following you'll find some requests that tests different functionalities of deegree WCS and the results that should be delivered. Testing of these requests with your web browser ensures that you installation works correctly.

request 1: Get service section from the WCS capabilities:

```
http://localhost:8080/deegree-wcs/services?
service=WCS&request=GetCapabilities&version=1.0.0&section=/WCS_Capab
ilities/Service
```

result:
```xml
<?xml version="1.0" encoding="UTF-8"?>
<Service updateSequence="1.0.0" version="1.0.0"
   xmlns="http://www.opengis.net/wcs"
   xmlns:deegree="http://www.deegree.org"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink">
   <metadataLink about="http://www.deegree.org" metadataType="TC211"
   xlink:href="http://www.deegree.org" xlink:title="deegree WCS metadata"
   xlink:type="simple"/>
   <description>deegree OGC WCS 1.0.0 reference implementation</description>
   <name>deegree WCS</name>
   <label>deegree WCS</label>
   <responsibleParty>
      <individualName>Andreas Poth</individualName>
      <organisationName>lat/lon</organisationName>
      <positionName>technical director</positionName>
      <contactInfo>
         <phone>
            <voice>12345678</voice>
            <voice>87654321</voice>
            <facsimile>656454534323</facsimile>
            <facsimile>31243647</facsimile>
         </phone>
         <address>
            <city/>
            <administrativeArea/>
            <postalCode/>
            <country/>
         </address>
         <onlineResource xlink:href="http://www.lat-lon.de"
                         xlink:type="simple"/>
      </contactInfo>
   </responsibleParty>
   <fees codeSpace="http://www.deegree.org">NONE</fees>
   <accessConstraints
         codeSpace="http://www.deegree.org">NONE</accessConstraints>
   <accessConstraints
         codeSpace="http://www.deegree.org">SOME</accessConstraints>
</Service>
```

request 2: Get description of one named coverage:

```
http://localhost:8080/deegree-wcs/services?
service=WCS&request=DescribeCoverage&version=1.0.0&coverage=saltlake
city
```

result:
```xml
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescription xmlns="http://www.opengis.net/wcs"
xmlns:deegree="http://www.deegree.org/wcs"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  version="1.0.0">

  <CoverageOffering>
    <name>saltlakecity</name>
    <label>saltlakecity</label>
```

```xml
      <lonLatEnvelope srsName="WGS84(DD)">
        <gml:pos dimension="2">-111.8998528924585 40.71941441838457</gml:pos>
        <gml:pos dimension="2">-111.80644636933437 40.828206328381825</gml:pos>
      </lonLatEnvelope>
      <keywords>
        <keyword />
      </keywords>
      <domainSet>
        <spatialDomain>
          <gml:Envelope srsName="EPSG:26912">
            <gml:pos dimension="2">424000.5 4507999.5</gml:pos>
            <gml:pos dimension="2">432000.5 4519999.5</gml:pos>
          </gml:Envelope>
        </spatialDomain>
      </domainSet>
      <rangeSet>
        <RangeSet>
          <name>saltlakecity</name>
          <label>saltlakecity</label>
          <nullValues>
            <interval atomic="false">
              <min>-99</min>
              <max>-99</max>
              <res>1</res>
            </interval>
            <singleValue>-9999</singleValue>
          </nullValues>
        </RangeSet>
      </rangeSet>
      <supportedCRSs>
        <requestResponseCRSs>EPSG:26912</requestResponseCRSs>
        <requestCRSs>EPSG:4326 EPSG:26912</requestCRSs>
        <nativeCRSs>EPSG:26912</nativeCRSs>
      </supportedCRSs>
      <supportedFormats>
        <formats>jpeg</formats>
        <formats>GeoTiff</formats>
        <formats>png</formats>
      </supportedFormats>
      <supportedInterpolations default="nearest neighbor">
        <interpolationMethod>nearest neighbor</interpolationMethod>
      </supportedInterpolations>
    </CoverageOffering>
</CoverageDescription>
```

request 3: Get description of all available coverages:

```
http://localhost:8080/deegree-wcs/services?
service=WCS&request=DescribeCoverage&version=1.0.0
```

result:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescription xmlns="http://www.opengis.net/wcs"
xmlns:deegree="http://www.deegree.org/wcs"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  version="1.0.0">

  <CoverageOffering>
    <name>saltlakecity</name>
    <label>saltlakecity</label>
    <lonLatEnvelope srsName="WGS84(DD)">
      <gml:pos dimension="2">-111.8998528924585 40.71941441838457</gml:pos>
      <gml:pos dimension="2">-111.80644636933437 40.828206328381825</gml:pos>
```

```xml
        </lonLatEnvelope>
        <keywords>
          <keyword />
        </keywords>
        <domainSet>
          <spatialDomain>
            <gml:Envelope srsName="EPSG:26912">
              <gml:pos dimension="2">424000.5 4507999.5</gml:pos>
              <gml:pos dimension="2">432000.5 4519999.5</gml:pos>
            </gml:Envelope>
          </spatialDomain>
        </domainSet>
        <rangeSet>
          <RangeSet>
            <name>saltlakecity</name>
            <label>saltlakecity</label>
            <nullValues>
              <interval atomic="false">
                <min>-99</min>
                <max>-99</max>
                <res>1</res>
              </interval>
              <singleValue>-9999</singleValue>
            </nullValues>
          </RangeSet>
        </rangeSet>
        <supportedCRSs>
          <requestResponseCRSs>EPSG:26912</requestResponseCRSs>
          <requestCRSs>EPSG:4326 EPSG:26912</requestCRSs>
          <nativeCRSs>EPSG:26912</nativeCRSs>
        </supportedCRSs>
        <supportedFormats>
          <formats>jpeg</formats>
          <formats>GeoTiff</formats>
          <formats>png</formats>
        </supportedFormats>
        <supportedInterpolations default="nearest neighbor">
          <interpolationMethod>nearest neighbor</interpolationMethod>
        </supportedInterpolations>
    </CoverageOffering>
</CoverageDescription>
```

request 4: Get a named coverage:
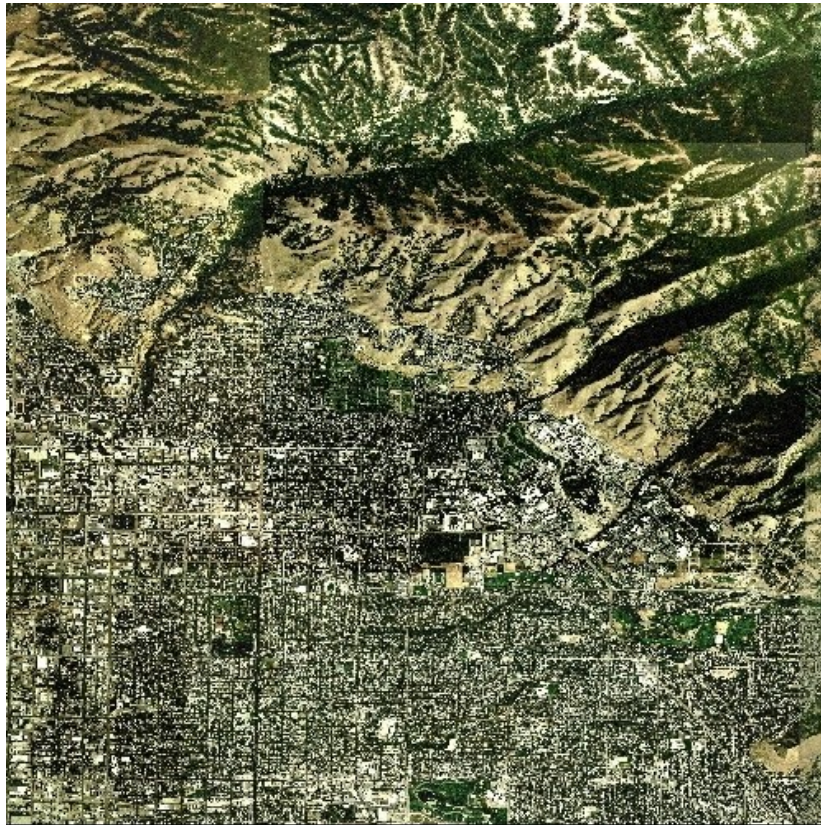
```
http://localhost:8080/deegree-wcs/services?
service=WCS&request=GetCoverage&version=1.0.0&coverage=saltlakecity&
CRS=EPSG:26912&response_CRS=EPSG:26912&BBOX=424000.5,4507999.5,43200
0.5,4519999.5&Width=500&height=500&format=jpeg
```

result:

Figure 1: Result of the example GetCoverage request

Since v2.3 all these example requests are available at the deegree WCS demo pages.

3 Architecture

Figure 2 shows the principle architecture of the modules involved in deegree WCS. The Web interface is realized by a servlet that has to be registered into a servlet engine like Tomcat or Jetty. The servlet chooses a handler class depending on the incoming request that delegates a request to the responsible service (in this case WCService). Depending on the requested coverage the WCService decides which kind of datasource has to be read and which (Grid-)CoverageReader is responsible for this. The decision about the (Grid-)CoverageReader to be used is made using the extension of the file(s) assigned to a coverage. At the moment image formats *.tif, *gif, *.bmp, *png and *.jpeg are supported. Also supported are GeoTIFF containing raw data (no color palette) and ECW format.

Delivering the result of a request the WCService re-calls the handler class which writes the result back to the servlet output stream. For performing a DescribeCoverage request the request processing is similar except that no difference is between different coverage sources.
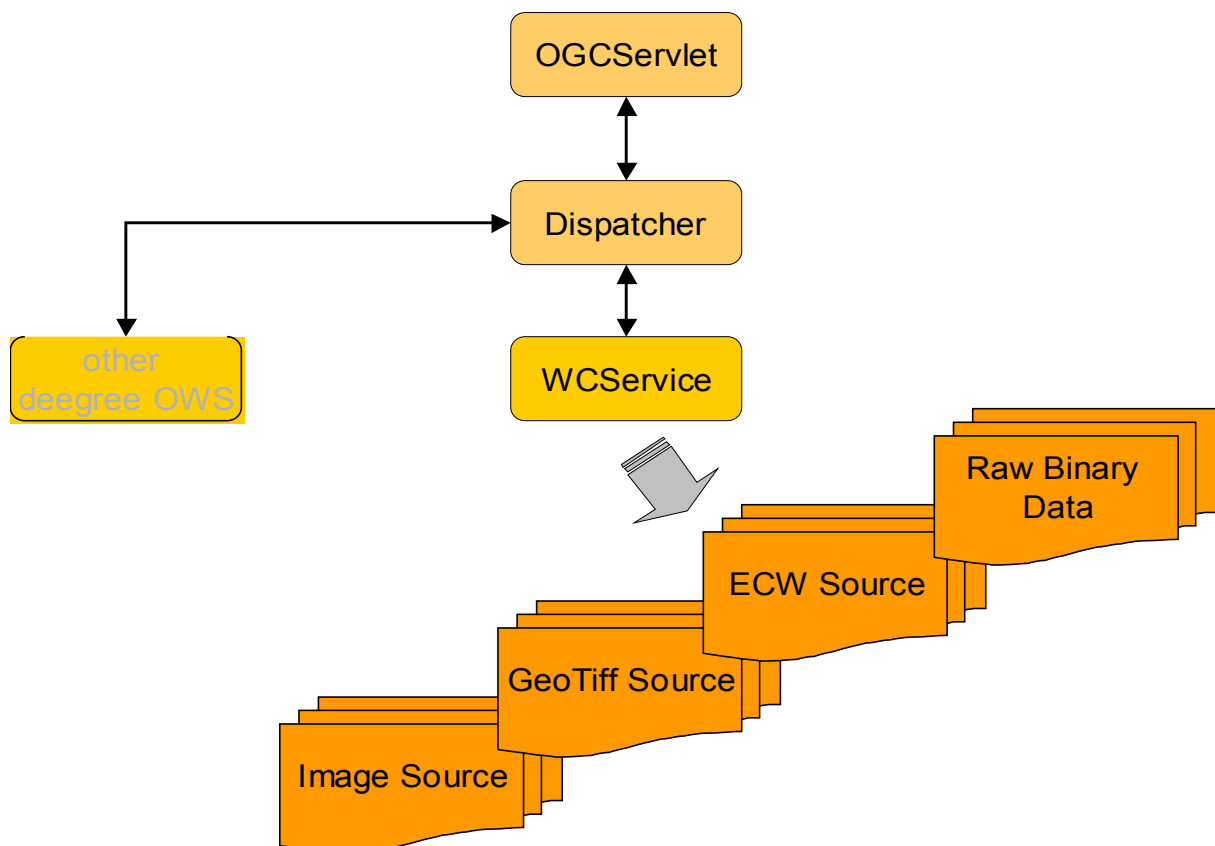

Figure 2: deegree WCS architecture overview

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:
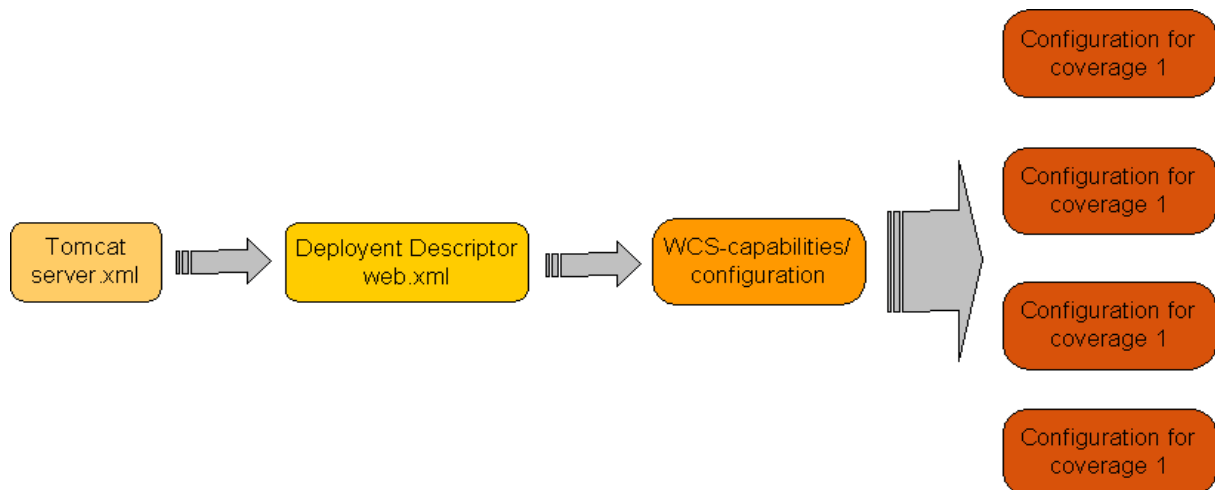

Figure 3: File dependencies for deegree WCS configuration

The central configuration document of the deegree WCS is based on a WCS capabilities document () enhanced by a few tags and with most elements defined optional. If a required value is missing it will be inserted automatically using default values → see below. Because of this, deegree WCS just like WMS supports a 'full configuration' where the user is able to set each single tag to a specific value and a 'brief configuration' where most of the tags can be left out.

The basic configuration allows usage of the full set of parameters that can be used for controlling the deegree WCS. This includes the parameters mentioned in the OGC WCS 1.0.0 specification for the Capabilities document, but also additional technical parameters as for example references to data source descriptions for each coverage or the maximum cache size.

In the following, the elements of the configuration file will be described in detail. Appendix A (Example WCS configuration/capabilities document) includes an example of a complete configuration/capabilities document.

4.2 The deegree WCS configuration document

4.2.1 deegree-Parameters

Within the root element some deegree-WCS-specific parameters are defined first. The <DefaultOnlineResource> is the URL by which the WCS operations can be

invoked. This parameter can be overwritten by the URLs defined in the request-definitions. This parameter is mandatory.

```xml
<deegree:deegreeParam>
  <!--
    The DefaultOnlineResource will be used if a required OnlineResource is
    not defined
  -->
  <deegree:DefaultOnlineResource xlink:href="http://localhost:8080/deegree-wcs/services"
    xlink:type="simple" />
  <!-- optional; default = 100 (MB) -->
  <deegree:CacheSize>250</deegree:CacheSize>
  <!-- maximum time for the execution of a request until an exception of time-
exceed is thrown.
    optional; default 5 Minutes -->
  <deegree:RequestTimeLimit>10</deegree:RequestTimeLimit>
  <!--
    list of directories to be scanned for coverages to be served by a WCS.
    deegree will look for configuration files in this directories and add the
    corresponding coverages  to the ContentMetadata section if not already
    present.optional; default: $RootDirectory$/WEB-INF/data
  -->
  <!-- Fature not yet implemented but follows very soon -->
  <deegree:DataDirectoryList>
    <deegree:DataDirectory>.</deegree:DataDirectory>
  </deegree:DataDirectoryList>

</deegree:deegreeParam>
```

The <CacheSize> parameter defines the size of the cache available to deegree-WCS in megabyte (this does not affect the cache for specific data sources that are are defined independently). This parameter is optional, its default value is 100 MB. By <RequestTimeLimit> the maximum time span is defined after which a request has to be processed. If this value is exceeded the processing is canceled and an exception will be thrown. Its default value is 5 minutes.

The <DataDirectoryList> element defines a list of directory names that will be searched for coverage description (configuration) documents. If a document is found, containing a description of a coverage that is not already registered the coverage will be added to the ContentMetadata section of the configuration/capabilities document automatically (***this feature is not implemented yet but will be in near future***).

4.2.2 Service Parameter

The following definition of service meta data is adopted from the WCS 1.0.0 specification. For detailed description of the contained elements and attributes please have a look to the WCS 1.0.0 specification.

```xml
<Service>
  <!--
    optional; no default
  -->
  <metadataLink about="http://www.deegree.org"
gml:remoteSchema="http://www.deegree.org"
    metadataType="TC211" xlink:actuate="onLoad"
    xlink:arcrole="http://www.deegree.org"
```

```xml
          xlink:href="http://www.deegree.org" xlink:role="http://www.deegree.org"
          xlink:show="new"
          xlink:title="deegree WCS metadata" xlink:type="simple" />
<!--
    optional; no default
-->
 <description>deegree WCS being OGC WCS 1.0.0 reference
 implementation</description>
<!--
    mandatory; if missing 'deegreewcs' will be used as default
-->
<name>deegree WCS</name>
<!--
    mandatory; if missing 'deegreewcs' will be used as default
-->
<label>deegree WCS</label>
<!--
    optional; no default
-->
<keywords>
    <keyword>deegree</keyword>
    <keyword>WCS</keyword>
    <type codeSpace="http://www.deegree.org">deegree internal</type>
</keywords>
<keywords>
    <keyword>reference implemenation</keyword>
    <keyword>WCS</keyword>
    <type codeSpace="http://www.deegree.org">OGC</type>
</keywords>
<!--
    optional; no default
-->
<responsibleParty>
    <!--
        mandatory; if missing 'deegree' will be used as default
    -->
    <individualName>Andreas Poth</individualName>
    <!--
        optional; no default
    -->
    <organisationName>lat/lon</organisationName>
    <!--
        optional; no default
    -->
    <positionName>technical director</positionName>
    <!--
        optional; no default
        if contactInfo is defined all sub-elements are are also optional
    -->
    <contactInfo>
      <phone>
        <voice>12345678</voice>
        <voice>87654321</voice>
        <facsimile>656454534323</facsimile>
        <facsimile>31243647</facsimile>
      </phone>
      <address>
        <deliveryPoint>Meckenheimer Allee 176</deliveryPoint>
        <deliveryPoint>Bonner Talweg</deliveryPoint>
        <city>Bonn</city>
        <administrativeArea>NRW</administrativeArea>
        <postalCode>53115</postalCode>
        <country>Germany</country>
        <electronicMailAddress>poth@lat-lon.de</electronicMailAddress>
        <electronicMailAddress>info@lat-lon.de</electronicMailAddress>
      </address>
      <onlineResource xlink:actuate="onLoad" xlink:arcrole="http://www.lat-
      lon.de"
```

```
      xlink:href="http://localhost:8080/deegree-wcs/services"
      xlink:role="http://www.lat-lon.de" xlink:show="new"
      xlink:title="lat/lon homepage" xlink:type="simple" />
    </contactInfo>
  </responsibleParty>
  <!--
    mandatory; if missing 'NONE' will be used as default
  -->
  <fees codeSpace="http://www.deegree.org">NONE</fees>
  <!--
    mandatory; if missing 'NONE' will be used as default
  -->
  <accessConstraints codeSpace="http://www.deegree.org">NONE</accessConstraints>
  <accessConstraints codeSpace="http://www.deegree.org">SOME</accessConstraints>
</Service>
```

4.2.3 Capability/Request-Parameter

The request-definition (referring to the operations GetCapabilities, GetCoverage and DescribeCoverage) is in its form identical to the definitions in OGC WCS 1.0.0 specification. However, some of the mandatory elements are declared optional. In case they are not supplied, deegree uses the corresponding default values. It is for example possible to not reference any of the requests explicitly (↓→ brief configuration).

The following example is used to explain this by means of a GetCapabilities request.

```
<GetCapabilities>
  <DCPType>
    <HTTP>
      <Get>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Get>
      <Post>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Post>
    </HTTP>
  </DCPType>
</GetCapabilities>
```

You have to adjust the "xlink:href" attribute of the Http Get <OnlineResource> element to your system.

4.2.4 Exceptions

In this section the `Exception` format can be set. The default format is `application/vnd.ogc.se_xml`

```
<Exception>
  <Format>application/vnd.ogc.se_xml</Format>
  <Format>application/deegree_xml</Format>
```

```
</Exception>
```

4.2.5 ContentMetadata (available coverages)

The ContentMetadata section of the capabilities/configuration document contains basic information about the coverages offered by a WCS like their names and bounding boxes. For each available coverage a <CoverageOfferingBrief> section is included. For a detailed description of all available elements please have a look at the OGC WCS 1.0.0 specification. The deegree WCS configuration extents the <CoverageOfferingBrief> sections by adding references to a detailed description of a coverage.

```
<!--
    mandatory; if missing it will be created by deegree and filled with
'CoverageOfferingBrief' descriptions for all coverages that can be found in
directories listed in 'DataDirectoryList'. If 'ContentMetadata' isn't defined or is
empty and no coverages that can be found in directories listed in
'DataDirectoryList' the configuration is invalid because a (deegree) WCS at least
have to serve one coverage. If 'ContentMetadata' is defined deegree adds all
coverages found in directories listed in 'DataDirectoryList' that are not defined
ContentMetadata automaticly. All attributes of 'ContentMetadata' are optional and
don't have a default
  -->
  <wcs:ContentMetadata xmlns:wcs="http://www.opengis.net/wcs">
    <CoverageOfferingBrief gml:id="ID000001">
      <description></description>
      <name>saltlakecity</name>
      <label />
      <lonLatEnvelope xmlns:wcs="http://www.opengis.net/wcs" srsName="WGS84(DD)">
        <gml:pos dimension="2">-111.89985862992859,40.71940082135575</gml:pos>
        <gml:pos dimension="2">-111.80644038553243,40.82820182642026</gml:pos>
      </lonLatEnvelope>
      <wcs:keywords>
        <wcs:keyword></wcs:keyword>
      </wcs:keywords>
      <deegree:Configuration>
        ../../data/utah/raster/saltlakecity/wcs_saltlakecity_configuration.xml
      </deegree:Configuration>
    </CoverageOfferingBrief>
  </wcs:ContentMetadata>
```

Instead of using absolute pathnames, configuration documents can also be referenced by relative pathnames starting at the directory where the capabilities document is stored.

The element <deegree:Configuration> references a document that contains detailed information regarding the coverage it is assigned to. Because these configuration documents may contain descriptions for more than one coverage the suitable description will be identified by its name. **The same name as defined in <CoverageOfferingBrief> must be used!**

<ContentMetadata> is the last section in the wcs_configuration.xml. Detailed information about the coverages are referenced for each coverage. So let's have a look at this document in the following chapter.

4.3 CoverageDescription

In this demo service configuration this document is located under $demo_wcs$/WEB-INF/data/utah/raster/dop/wcs_saltlakesatellite_configuration.xml where the directory dop contains all relevant data and configuration files needed for coverage saltlakecity. Basically this document for detailed description of a coverage consists of

1. a section based on result documents to an OGC WCS DescribeCoverage request and

2. an <Extension> defined in the 'deegree' WCS namespace which is used for describing access to the coverages (file-)resources.

For general description of the CoverageDescription please consult the OGC WCS 1.0.0 specification.

If you use the RasterTreeBuilder, this document will be generated automatically. Most of the configuration should be left untouched but still it might be useful to do some changes.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescription xmlns="http://www.opengis.net/wcs"
xmlns:deegree="http://www.deegree.org/wcs"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  updateSequence="String" version="1.0.0">
  <CoverageOffering>
    <description xmlns:wcs="http://www.opengis.net/wcs"></description>
    <name>saltlakecity</name>
    <label xmlns:wcs="http://www.opengis.net/wcs">saltlakecity</label>
    <lonLatEnvelope xmlns:wcs="http://www.opengis.net/wcs" srsName="WGS84(DD)">
      <gml:pos dimension="2">-111.8998528924585,40.71941441838457</gml:pos>
      <gml:pos dimension="2">-111.80644636933437,40.828206328381825</gml:pos>
    </lonLatEnvelope>
    <wcs:keywords xmlns:wcs="http://www.opengis.net/wcs">
      <wcs:keyword></wcs:keyword>
    </wcs:keywords>
    <domainSet>
      <spatialDomain xmlns:wcs="http://www.opengis.net/wcs">
        <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#26912">
          <gml:pos dimension="2">424000.5,4507999.5</gml:pos>
          <gml:pos dimension="2">432000.5,4519999.5</gml:pos>
        </gml:Envelope>
      </spatialDomain>
    </domainSet>
    <rangeSet>
      <RangeSet refSys="http://www.deegree.org" refSysLabel="String"
        semantic="http://www.deegree.org">
        <description xmlns:wcs="http://www.opengis.net/wcs"></description>
        <name>saltlakecity</name>
        <label xmlns:wcs="http://www.opengis.net/wcs">saltlakecity</label>
        <nullValues semantic="http://www.deegree.org" type="xs:integer">
        <interval atomic="false" closure="closed"
semantic="http://www.deegree.org">
            <min>-99</min>
            <max>0</max>
            <res>1</res>
          </interval>
          <singleValue>-9999</singleValue>
```

```
        </nullValues>
      </RangeSet>
    </rangeSet>
    <supportedCRSs>
      <requestCRSs xmlns:wcs="http://www.opengis.net/wcs">EPSG:4326 EPSG:26912
</requestCRSs>
      <requestResponseCRSs xmlns:wcs="http://www.opengis.net/wcs">EPSG:26912
</requestResponseCRSs>
      <nativeCRSs xmlns:wcs="http://www.opengis.net/wcs">EPSG:26912</nativeCRSs>
    </supportedCRSs>
    <supportedFormats xmlns:wcs="http://www.opengis.net/wcs" nativeFormat="jpg">
      <formats>jpeg</formats>
      <formats>GeoTiff</formats>
      <formats>png</formats>
    </supportedFormats>
    <supportedInterpolations default="nearest neighbor">
      <interpolationMethod>nearest neighbor</interpolationMethod>
    </supportedInterpolations>

    <deegree:Extension>
    ...
    </deegree:Extension>

  </CoverageOffering>
</CoverageDescription>
```

In the <requestResponseCRSs> you are able to add additional CRSs in future, to offer them with the WCS. At the moment you can only request the native CRS.

Furthermore the <supportedFormats> can be defined here.

The <deegree:Extension> element contains detailed descriptions about what kind of resource a coverage is stored at and where it can be found. The basic concept of deegree WCS for (grid) coverage management is to split large coverages into several tiles with an approximate size between 300x300px and 600x600px and to pre-calculate different resolution versions for each coverage arranged in a **pyramidal data structure as raster tree**. deegree WCS knows more than one way to store this information and descriptions of the resources location as well as its type have to be given.
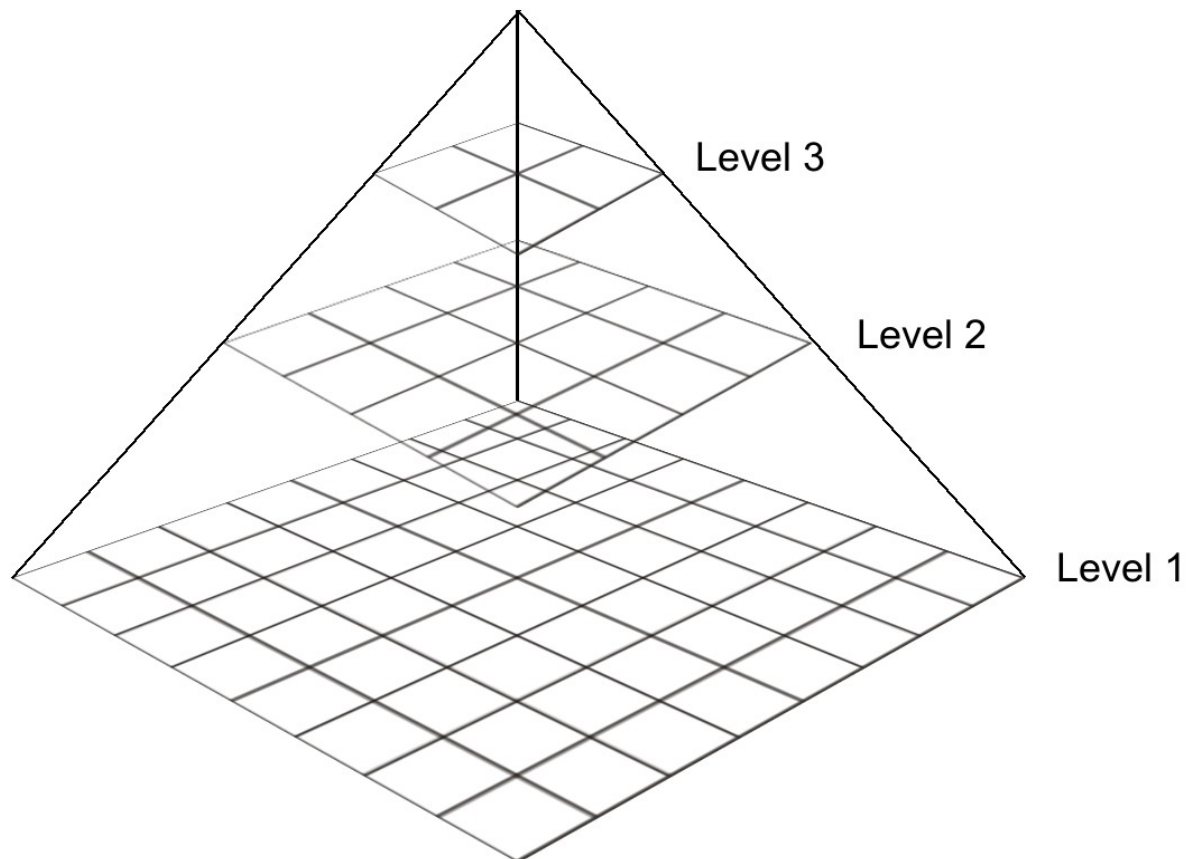
Figure 4: deegree WCS pyramidal data structure

deegree WCS knows three different kinds of resources:

1. *file*: the coverage is stored in one or more files each assigned to one bounding box and one spatial resolution level.

2. *shapeIndexed*: each spatial resolution level is stored within several (possibly several thousands) of tiles (data/image files). The tiles structure is described by one shapefile for each resolution level.

3. *nameIndexed*: similar to *fileIndexed* except that the geometric structure (extent) of the tiles is mapped to the file names

4. Support for databases, e.g. Oracle 10g Georaster

A description for creating *shapeIndexed* and *fileIndexed* coverages from one or more large source files can be found in Chapter 4.

4.3.1 File coverage source

The following XML-fragment shows an example for a CoverageDescription extension describing access to a coverage available in two files; one for the southern and one for the northern hemisphere.

```
<deegree:Extension type="file">
   <deegree:Resolution max="99999999" min="0">
      <deegree:File height="1250" width="2500">
         <deegree:Name>d:/data/raster/mlatlon_south.tif</deegree:Name>
         <gml:Envelope
            srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:coordinates>-180,-90 180,0</gml:coordinates>
         </gml:Envelope>
      </deegree:File>
      <deegree:File height="1250" width="2500">
         <deegree:Name>d:/data/raster/mlatlon_north.tif</deegree:Name>
         <gml:Envelope
            srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:coordinates>-180,0 180,00</gml:coordinates>
         </gml:Envelope>
      </deegree:File>
   </deegree:Resolution>
</deegree:Extension>
```

The attribute 'type' of the Extension element declares that the following XML
fragment describes a file based coverage access. The first element,
<Resolution>, holds two attributes defining the spatial resolution range the
following file descriptions are valid for. The range is given in units of the
coverages native CRS. In the example just one Resolution element is defined
matching the complete range of useful Resolutions. In principle there is no
limitation for splitting a coverage into several numbers of distinct resolution
levels. In this case for each range one resolution element must be defined. Each
Resolution element contains one or more File elements describing the extent and
access to one coverage file. Valid file formats are described above.

You won't find this example in the demo service.

4.3.2 shapeIndexed coverage source

The configuration for shapeIndexed coverage sources is very similar to the
type="file" based one. The major difference is that a shapefile
(type="shapeIndexed") instead of a coverage source file is referenced. Notice
that the name of the shape must be set without file extension.

```
<deegree:Extension xmlns:wcs="http://www.opengis.net/wcs" type="shapeIndexed">
  <deegree:Resolution max="99999999" min="16.0">
    <deegree:Range>
      <deegree:Name>default</deegree:Name>
    </deegree:Range>
    <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh16.0</deegree:Shape>
  </deegree:Resolution>
  <deegree:Resolution max="16.0" min="8.0">
    <deegree:Range>
      <deegree:Name>default</deegree:Name>
    </deegree:Range>
    <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh8.0</deegree:Shape>
  </deegree:Resolution>
  <deegree:Resolution max="8.0" min="4.0">
    <deegree:Range>
      <deegree:Name>default</deegree:Name>
```

```xml
    </deegree:Range>
    <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh4.0</deegree:Shape>
  </deegree:Resolution>
  <deegree:Resolution max="4.0" min="2.0">
    <deegree:Range>
      <deegree:Name>default</deegree:Name>
    </deegree:Range>
    <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh2.0</deegree:Shape>
  </deegree:Resolution>
  <deegree:Resolution max="2.0" min="0.0">
    <deegree:Range>
      <deegree:Name>default</deegree:Name>
    </deegree:Range>
    <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh1.0</deegree:Shape>
  </deegree:Resolution>
</deegree:Extension>
```
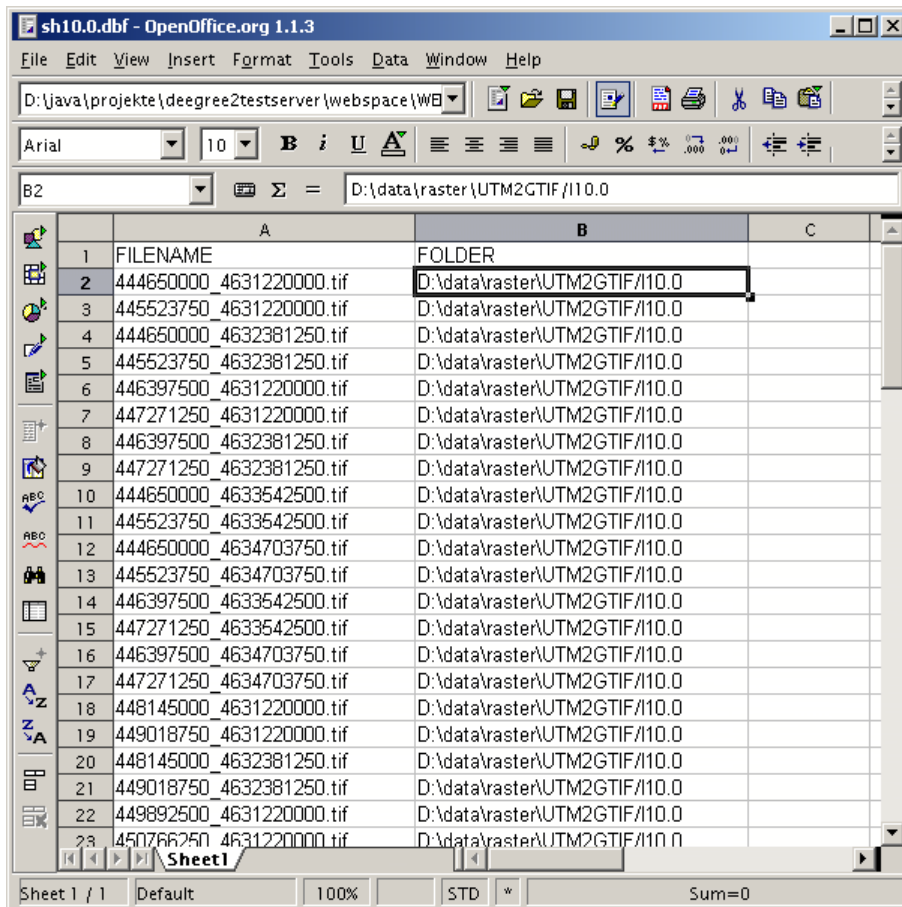
ShapeFiles also can be referenced by relative paths starting at the directory where the configuration file is stored.

The shapefile contains spatial extent and storage location for each tile (file) assigned to the coverage resolution level. The attributes '*tileProperty*' and '*directoryProperty*' name the columns of the DBase file assigned to the shapefile holding the tile names and the name of the directory(ies) where they are stored. The '*srsName*' attribute defines the EPSG name/code for the coordinate reference system the coverage sources are stored in.

The Resolution element contains an additional element named <Range>. This will be used in future releases of the deegree WCS to define the range(s) (e.g. time, extent, spectral range ...) a coverage source is assigned to. **At the moment only the default range is supported.**

Figure 5: Example for a DBase file containing references to the tiles assigned to a coverage level

4.3.3 Database Support – Oracle 10g Georaster

The following example shows, how to configure an Oracle Georaster database connection within a coverage description.

```
<!--
'file' storages are similar to 'nameIndexed' storages except that not
directories are
referenced but single files or groups of files.
-->
<deegree:Extension type="OracleGeoRaster">
    <deegree:Resolution max="99999999" min="1">
        <deegree:Connection>
            <jdbc:JDBCConnection xmlns:jdbc="http://www.deegree.org/jdbc">
                <jdbc:Driver>oracle.jdbc.OracleDriver</jdbc:Driver>
                <jdbc:Url>jdbc:oracle:thin:@localhost:1521:devs</jdbc:Url>
                <jdbc:User>dev</jdbc:User>
                <jdbc:Password>dev</jdbc:Password>
                <jdbc:SecurityConstraints/>
                <jdbc:Encoding>iso-8859-1</jdbc:Encoding>
            </jdbc:JDBCConnection>
            <!-- table containin raster -->
            <deegree:Name>MyTable</deegree:Name>
            <!-- column containing raster -->
            <deegree:Column>MyColumn</deegree:Column>
        </deegree:Connection>
```

```
    </deegree:Resolution>
</deegree:Extension>
```

4.3.4 Dynamicly integrated external datasources

A very flexible way to integrate raster data into deegree WCS is usage of dynamic datasources. Using a dynamic datasource deegree WCS does not know anything about the rasters physical backend. On interesting option is to used this kind of datasource to generate raster on the fly e.g. by reading a set of points from a WFS or a database for requested boundingbox and to interpolate them into a raster.

To use dynamic datasources following configuration must be done within according coverage description (configuration) document:

```
<deegree:Extension xmlns:wcs="http://www.opengis.net/wcs" type="script">
   <deegree:Resolution max="99999999" min="0">
      <deegree:Script>
         <deegree:Name>de.latlon.interpolation.WPSInterpolation</deegree:Name>
         <deegree:Parameter>BBOX</deegree:Parameter>
         <deegree:Parameter>FORMAT</deegree:Parameter>
         <deegree:Parameter>SIZE</deegree:Parameter>
         <deegree:ResultFormat>jpeg</deegree:ResultFormat>
         <deegree:StorageLocation>./temp</deegree:StorageLocation>
      </deegree:Script>
   </deegree:Resolution>
</deegree:Extension>
```

The <Script> Element does expect a set parameters to enable a dynamic datasource creating a raster that matches conditions/parameters of a GetCoverage request. First and most important is the name of a Java class that offeres/creates a raster. As you see valid classes must not be part of deegree; the only rule they must fullfill is to implement inerface org.deegree.ogcwebservices.wcs.ExternalDataAccess. After setting the class name a list of optional parameters (<Parameter>) are defined that will be passed to defined Java class. These are more or less place holders for future implementation and should be defined like in example above. The last two elements defines the format of the raster returned by a dynamic datasource and where it is stored. This means, the implementing Java class does not return the raster itself but just the name of the resource/location where the raster is stored.

5 Creating coverage tiles

deegree offers one tool for creating optimized data structures from one or more large image source(s) - RasterTreeBuilder (RTB).

Before starting to use RTB, you should take a few minutes to think of the use cases for your WCS and the coverages you are setting up and choose the configuration options accordingly. E.g. if you like to make topographic maps available through a WCS it doesn't make sense to create each resolution level that is technically possible as the optical quality of levels with much be less than half the original resolution is usually not acceptable (for more background information, have a look at „3.1.2 Calculation of one Coverage – Inside the RTB "). If you just want to deliver satellite images in their original resolution for further processing on workstations, a file based configuration will most likely be the best.

5.1.1 RasterTreeBuilder (RTB)

The main utility program coming with deegree WCS is the RasterTreeBuilder. It is located in org.deegree.tools.raster package and offers several configuration parameters to optimize data structure for fast access to coverage source(s).

The basic idea is that especially for large coverages in most cases a GetCoverage request targets just a small subset of the coverage's spatial extent with a high resolution or a large subset with a low resolution. So it will be useful to have a mechanism that avoids reading and processing data that is not required.

The deegree approach to solve this problem is similar to mechanisms used by several other software packages handling large amounts of raster data:

1. In step of a pre-processing, the original, large data sources will first be resampled into several levels of lower resolution.

2. After that each resolution level will be split into tiles each having a moderate size.

The optimal size strongly depends on the desired use of a WCS. E.g. if a WCS shall be used for online image visualization within a standard web browser a GetCoverage request hardly will use WIDTH and HEIGHT exceeding 800x800px. So an optimal size of a tile will be ~500x500px guaranteeing no more than nine tiles are needed to create the result to a GetCoverage request and in many cases even less tiles are needed.

5.1.2 Using RasterTreeBuilder

To resample and tile one or more source images the RasterTreeBuilder application needs definition of several parameters passed via command line

options. For each phase of the program – input, processing, output – there are several parameters needed:

Input

Precondition:
An image file passed to the RTB must have a world file assigned to it by having the same name but with extension *.tfw, *.wld, *.jgw, *.jpgw, *.gfw, *.gifw, *.pgw or *.pngw. GeoTIFF files are also accepted.

There are three alternative ways/parameters to define which input files shall be used for creating a raster tree:

1) `–mapFiles` defines a list of image file names (including full path information) separated by ',', ';' or '|'

2) `-rootDir` defines a directory that shall be parsed for files in a known image format. Each file found will be used as input.

`-subDirs` conditional parameter used with `-rootDir`. It defines if all sub directories of `-rootDir` shall be parsed too (true|false - default false)

3) `-dbaseFile` name of a dBase file that contains a column, listing all files to be considered by the program

`-fileColumn` name of the column containing the file names (mandatory if `-dbaseFile` is defined)

`-baseDir` name of the directory where the files are stored. If this parameter will not be set the program assumes the -fileColumn contains completely referenced file names (optional)

`-sortColumn` If `-dbaseFile` is defined one can define a column that shall be used for sorting the files referenced by the `-fileColumn` (optional)

`-sortDirection` If -sortColumn is defined this parameter will be used for definition of sorting direction (UP|DOWN default UP)

`-srs` name of the spatial reference system used for the coverage (default EPSG:4326)

Defines the EPSG-Code of the spatial reference system (SRS) that the coordinates of the input data are in. Just this native SRS is valid in the moment here.

! Be aware that the RTB is not able to project/transform your raster data in the moment !

`-worldFileType` two types of are common:

a) the boundingbox is defined on the center of the corner pixels

b) the boundingbox is defined on the outer corner of the corner pixels

Defaulting: a) is default and will be used if this parameter is not set; second will be used if '-worldFileType outter' is defined.

Processing:

`-noOfLevel` number of tree levels created (optional default = 1)

Number of levels of different spatial resolutions to be calculated starting from the basis to the peak of the pyramid (Look at the examples for getting to understand in detail how the RTB works).

`-outputFormat` name of the image format used for created tiles (png|jpg|jpeg| bmp|tif|tiff|gif default png)

`-quality` image quality if jpeg is used as output format; valid range is from 0..1 (default 0.95)

`-maxTileSize` maximum size of created raster tiles in pixel (default 500)

`-bbox` boundingbox of the the resulting coverage. If not set the bbox will be determined by analysing the input map files. (optional)

The bounding box coordinates must match the type of the spatial reference system flag (`-srs`).

If you set a bigger bounding box than covered by the input images, you can have regions within the same coverage with and without rasterdata (It is like islands in the ocean). For example: If your input data is spread like a patchwork. For setting the color of the non-covered areas (ocean), use the `bgColor`-flag.

`-resolution` spatial resolution of the coverage (at the basis of the pyramid). If not set, the resolution will determined by analysing the input map files. This parameter is conditional; if -bbox is defined -resolution must be defined, too.

In order to calculate the spatial resolution of the coverage, you need to know the distance from N-S, W-E and the corresponding number of pixels. So if you give RTB the bbox, give it the pixel size as well.

`-interpolation` interpolation method used for resampling raster images (Nearest Neighbor|Bicubic|Bilinear default Nearest Neighbor)

`-bgColor` defines the background color of the created tiles for those regions where no data is available

If no `-bgColor` is defined, a transparent background will be used for image formats that support transparency (e.g. png). Black is used for all other formats (e.g. bmp).

For example a '`-bgColor` 0xFFFFFF' defines a white background.

`-cacheSize` number of images that shall be cached.

During the processing, input images are read in several times for different steps in the processing phase. A larger value (~20) increases speed (and reduces I/O operations) but also amount of required memory. (default = 5)

Output

`-outDir` directory where resulting tiles and describing shape(s) will be stored (mandatory)

`-baseName` base name used for creating names of the raster tile files. It also will be the name of the created coverage. (mandatory)

`-capabilitiesFile` name of a deegree WCS capabilities/configuration file. If defined the program will add the created rastertree as a new coverage to the WCS configuration.

! **not implemented yet; Please add the configuration of your created coverage manually** !

Misc.

`-h` or `-?` print out the help

Example 1: Creating a coverage with 5 levels out of a source directory on a Windows platform

```
java -Xms300m -Xmx1000m -classpath
.;..\..\classes;..\..\lib\vecmath.jar;..\..\lib\log4j-
1.2.9.jar;..\..\lib\jai_codec.jar;..\..\lib\jai_core.jar;..\..\lib\mlibwrap
per_jai.jar;..\..\lib\jts-1.8.jar;..\..\lib\jaxen-1.1-beta-
8.jar;..\..\lib\acme.jar;..\..\lib\batik-awt-util.jar;..\..\lib\commons-
beanutils-1.5.jar;..\..\lib\commons-codec-1.3.jar;..\..\lib\commons-
collections-3.1.jar;..\..\lib\commons-digester-1.7.jar;..\..\lib\commons-
discovery-0.2.jar;..\..\lib\commons-
logging.jar;..\..\lib\j3dcore.jar;..\..\lib\j3dutils.jar;..\..\lib\axis.jar
;..\..\lib\ehcache-1.2.0_03.jar;..\..\lib\deegree2.jar
org.deegree.tools.raster.RasterTreeBuilder -outDir
../../data/utah/raster/saltlakecity -baseName saltlakecity -outputFormat
jpg -maxTileSize 500 -noOfLevel 5 -srs EPSG:26912 -rootDir
../../data/utah/raster/orig
```

Snipplet/Flag	Explanation
`java –Xms300m –Xmx1000m`	The Java Virtual Machine is started with a minimum memory of 300 MB and a maximum of 1000 MB

`-classpath .;`
`../../lib/deegree2.jar;`
`../../lib/acme.jar;`
`../../lib/commons-beanutils-1.5.jar;`
`../../lib/commons-codec-1.3.jar;`
`../../lib/commons-collections-`
`3.1.jar;`
`../../lib/commons-digester-1.7.jar;`
`../../lib/commons-discovery-0.2.jar;`
`../../lib/commonslogging.jar;`
`../../lib/jai_codec.jar;`
`../../lib/jai_core.jar;`
`../../lib/mlibwrapper_jai.jar;`
`../../lib/j3dcore.jar;`
`../../lib/j3dutils.jar;`
`../../lib/vecmath.jar;`
`../../lib/jts-1.8.jar;`
`../../lib/log4j-1.2.9.jar;`
`../../lib/axis.jar;`
`../../lib/jaxen-1.1-beta-8.jar`
	Windows Java Classpath (delimited by ';'): Registering all needed libraries for starting the RasterTreeBuilder

`org.deegree.tools.raster.RasterTreeBuilder`	the RTB program
`-rootDir d:/data/raster/europe/`	Input directory, where sources images can be found
`-outDir e:/temp`	Output directory, where result coverages is written
`-outputFormat jpg`	Output format
`-quality 0.95`	Compression Level for JPG, 95%
`-srs EPSG:26912`	Spatial reference system the coordinates of the sources are in.
`-noOfLevel 5`	Number of levels to be calculated
`–baseName MyCoverage`	Name of the Coverage

Table 2: RasterTreeBuilder command line parameters (Windows)

Example 2: Creating a coverage with 4 Levels out of a dBase file on LINUX platform

```
java -Xms300m -Xmx1000m
-classpath.:../../lib/deegree2.jar:../../lib/acme.jar:../../lib/commons-
beanutils-1.5.jar:../../lib/commons-codec-1.3.jar:../../lib/commons-
collections-3.1.jar:../../lib/commons-digester-1.7.jar:../../lib/commons-
discovery-
0.2.jar:../../lib/commonslogging.jar:../../lib/jai_codec.jar:../../lib/jai_
core.jar:../../lib/mlibwrapper_jai.jar:../../lib/j3dcore.jar:../../lib/j3du
tils.jar:../../lib/vecmath.jar:../../lib/jts-1.8.jar:../../lib/log4j-
1.2.9.jar:../../lib/axis.jar:../../lib/jaxen-1.1-beta-8.jar
org.deegree.tools.raster.RasterTreeBuilder -dbaseFile
/home/deegree/input.dbf -fileColumn LOCATION -sortColumn DATE
-sortDirection DOWN -noOfLevel 4 -interpolation "Nearest Neighbor"  -bbox
2573388.0,5618175.0,2574150.03,5618416.98686 -srs EPSG:31466 -resolution
0.09 -outDir /home/deegree/output -outputFormat png -bgColor 0xFFFF00
-baseName MyCoverage
```

Snipplet/Flag	*Explanation*
`java -Xms300m -Xmx1000m`	The Java Virtual Machine is started with a minimum memory of 300 MB and a maximum of 1000 MB

`-classpath .:`
`../../lib/deegree2.jar:`
`../../lib/acme.jar:`
`../../lib/commons-beanutils-1.5.jar:`
`../../lib/commons-codec-1.3.jar:`
`../../lib/commons-collections-3.1.jar:`
`../../lib/commons-digester-1.7.jar:`
`../../lib/commons-discovery-0.2.jar:`
`../../lib/commonslogging.jar:`
`../../lib/jai_codec.jar:`
`../../lib/jai_core.jar:`
`../../lib/mlibwrapper_jai.jar:`
`../../lib/j3dcore.jar:`
`../../lib/j3dutils.jar:`
`../../lib/vecmath.jar:`
`../../lib/jts-1.8.jar:`
`../../lib/log4j-1.2.9.jar:`
`../../lib/axis.jar:`
`../../lib/jaxen-1.1-beta-8.jar`
	Linux Java Classpath (delimited by ':'): Registering all needed libraries for starting the RasterTreeBuilder

`org.deegree.tools.raster.RasterTreeBuilder`	the RTB program
`-dbaseFile /home/deegree/input.dbf`	dBase-File referencing the input images

Snipplet/Flag	Explanation
-fileColumn LOCATION	Column with absolute path names to each input image
-sortColumn DATE	Column in the dBase-file used for sorting
-sortDirection DOWN	Sorting the table based on the sortColumn descending
-noOfLevel 4	Number of levels to be calculated
-interpolation "Nearest Neighbor"	Nearest Neighbor algorithm is used (Be aware to use ' " ' for the entire expression!)
-bbox 2573388.0,5618175.0,2574150.03,5618416.98686	Setting a bounding box bigger than the area covered by the input images.
-srs EPSG:31466	Spatial reference system the coordinates of the sources are in.
-resolution 0.09	Spatial resolution of 0.09 m per pixel (FYI: bbox is in metric coordinates, EPSG:31466)
-outDir /home/deegree/output	Output directory, where result coverages is written
-outputFormat png	Output format
-bgColor 0xFFFF00	Setting the background color to yellow for regions not covered by input images.
-baseName MyCoverage	Name of the Coverage

Table 3: RasterTreeBuilder command line parameters (Linux)

5.1.3 Calculation of one Coverage – Inside the RTB

To use the parameters correctly you should understand how RTB works and how the number of resolution levels and tile size is coupled.

Imagine a source image of 4000x4000px with a spatial resolution of 1x1m per pixel. Use '-noOfLevel 3' to force the creation of three resolution levels having 1m (base level of the pyramid), 2m and 4m (peak of the pyramid) spatial resolution. For each level, one tiling step will be performed on the highest resolution level (1m). At the first tiling step, an image is split into four smaller images.

In our example in a first step, the source image is split in tiles each having 2000x2000px size; in the second step each of this files is split into four tiles each 1000x1000px and in the last, third step each of this tiles is split again into four tiles each 500x500px size (as stated above, this is a good size for normal visualization use cases). So you will receive $4^3 = 64$ tiles in total for the highest resolution level.

For the second highest resolution level the original image will be resampled to a spatial resolution of 2m (double the highest resolution). After this, two tiling steps will be performed . We will receive $4^2 = 16$ tiles of 500x500px size for the second highest resolution level.

To create the third level the image will be resampled again to a resolution of 4m. Just one tiling step will be performed resulting in four tiles of 500x500px size. The result is a tiled and pyramidal data structure like illustrated in Figure 4.

RTB will create a deegree CoverageDescription as XML document and a sub directory for each source image and resolution level within the defined output directory. Shapefiles containing the location of the tiles (used as an index) as described above will also be created.

5.1.4 Referencing the new generated coverage to the wcs_configuration.xml

Once you have run the RasterTreeBuilder successfully you need to link the wcs_configuration.xml file to the coverage description file. Within the `<wcs:ContentMetadata>` you need to add a new `<CoverageOfferingBrief...>` increasing the `gml:id="ID000001"` by one. Change the `<name>` and `<lonLatEnvelope>` to the one used in the coverage description. Finally set the (absolute/relativ) path to the coverage description in <deegree:Configuration>.

```
<!--
    mandatory; if missing it will be created by deegree and filled with
'CoverageOfferingBrief'
    descriptions for all coverages that can be found in directories listed in
'DataDirectoryList'.
    If 'ContentMetadata' isn't defined or is empty and no coverages that can be
found in
    directories listed in 'DataDirectoryList' the configuration is invalid because
a (deegree) WCS
    at least have to serve one coverage.
    If 'ContentMetadata' is defined deegree adds all coverages found in directories
listed in
    'DataDirectoryList' that are not defined ContentMetadata automaticly
    All attributes of 'ContentMetadata' are optional and don't have a default
  -->
  <wcs:ContentMetadata xmlns:wcs="http://www.opengis.net/wcs">
    <CoverageOfferingBrief gml:id="ID000001">
      <description></description>
      <name>saltlakecity</name>
      <label />
      <lonLatEnvelope xmlns:wcs="http://www.opengis.net/wcs" srsName="WGS84(DD)">
        <gml:pos dimension="2">-111.89985862992859,40.71940082135575</gml:pos>
        <gml:pos dimension="2">-111.80644038553243,40.82820182642026</gml:pos>
```

```
      </lonLatEnvelope>
      <wcs:keywords>
        <wcs:keyword></wcs:keyword>
      </wcs:keywords>
      <deegree:Configuration>
        ../../data/utah/raster/saltlakecity/wcs_saltlakecity_configuration.xml
      </deegree:Configuration>
    </CoverageOfferingBrief>
  </wcs:ContentMetadata>
```

Afterwards restart the tomcat and the coverage should be served.

6 Advanced configuration

6.1.1 Manual Tomcat configuration

The location of deegree's libraries and the central deegree configuration file `wcs_configuration.xml` has to point to the application server (in this case Apache Tomcat 5.5.x). Tomcat offers several possibilities to register and configure web contexts. An example based on one of these possibilities is the following.

The easiest way to register deegree web services with Tomcat is to copy the `deegree-wcs.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-wcs.war file (which is nothing more than a .zip file) to the webapps directory. The name of the .war sets the name of the service address:

http://localhost:8080/**deegree-wcs**

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

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

Afterwards Tomcat needs information about the root directory of the WMS. 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-wcs.xml, and fill it with the following information

```
<Context docBase="c:/deegree/webapps/deegree-wcs" path="/deegree-wcs">
</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-wcs/. For further information have a look at the Tomcat documentation included with the installation.

The name of the main directory is arbitrary whereas Tomcat definitely looks for a subdirectory in the root directory named WEB-INF (in capital letters – even on a Windows system). This directory was automatically created after unpacking the deegree WCS zip archive. Here the Deployment-Descriptor (`web.xml`) is located, that is analysed by Tomcat to identify the servlet(s) that belong to the application, their names, the parameters that are delivered to the servlet(s) and information about the existing access restrictions.

Before starting the deegree WCS the following data set entry in `web.xml` is essential:

```xml
<?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>
  <display-name>deegree 2.2</display-name>
  <description>deegree 2.2 OWS</description>
  <servlet>
    <servlet-name>owservice</servlet-name>
    <servlet-class>org.deegree.enterprise.servlet.OGCServletController</servlet-
class>
    <init-param>
      <param-name>services</param-name>
      <param-value>wcs</param-value>
      <description>list of supported services, e.g.: wfs,wms (comma separated)
      allways use lowercase</description>
    </init-param>

    <init-param>
      <param-name>wcs.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WCSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wcs.config</param-name>
      <param-value>WEB-INF/conf/wcs/wcs_configuration.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>owservice</servlet-name>
    <url-pattern>/services</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>

  <error-page>
    <error-code>500</error-code>
    <location>/error.jsp</location>
  </error-page>
  <error-page>
    <exception-type>org.deegree.ogcwebservices.OGCWebServiceException</exception-
    type>
    <location>/error.jsp</location>
  </error-page>
</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 chosen freely, but take care that the same name that is defined here is also used in the servlet-mapping. The servlet is located in the library deegree2.jar.

The tags `<init-param>` transfers parameters that is 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

WCS is defined to be available (other possible values at the moment are: WMS, WFS, SOS,WPVS,WAS,WSS and CS-W).

➢ 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 (WCS 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.deegree.enterprise.servlet.WCSHandler`' 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 OWS available through a context the web.xml looks like this (the example defines a WCS as well as a WMS):

```xml
<?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>
  <display-name>deegree 2.2</display-name>
  <description>deegree 2.2 OWS</description>
  <servlet>
    <servlet-name>owservice</servlet-name>
    <servlet-class>org.deegree.enterprise.servlet.OGCServletController</servlet-class>

    <init-param>
      <param-name>services</param-name>
      <param-value>wcs</param-value>
      <description>
        list of supported services, e.g.: wfs,wms (comma separated) allways use
        lowercase
      </description>
    </init-param>

    <!-- WMS INITIALIZING PARAMETERS -->
    <init-param>
      <param-name>wms.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WMSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wms.config</param-name>
      <param-value>WEB-INF/conf/wcs/wms_configuration.xml</param-value>
    </init-param>

    <!-- WFS INITIALIZING PARAMETERS -->
    <init-param>
      <param-name>wfs.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WFSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wfs.config</param-name>
      <param-value>WEB-INF/conf/wcs/LOCALWFS_capabilities.xml</param-value>
    </init-param>
```

```xml
  <!-- WCS INITIALIZING PARAMETERS -->

  <init-param>
    <param-name>wcs.handler</param-name>
    <param-value>org.deegree.enterprise.servlet.WCSHandler</param-value>
  </init-param>
  <init-param>
    <param-name>wcs.config</param-name>
    <param-value>WEB-INF/conf/wcs/wcs_configuration.xml</param-value>
  </init-param>

  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>owservice</servlet-name>
  <url-pattern>/services</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>/index.jsp</welcome-file>
</welcome-file-list>

<error-page>
  <error-code>500</error-code>
  <location>/error.jsp</location>
</error-page>

<error-page>
  <exception-type>org.deegree.ogcwebservices.OGCWebServiceException</exception-type>
  <location>/error.jsp</location>
</error-page>

</web-app>
```

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-wcs.xml and respectively the name of the .war which you deployed in the $TOMCAT_HOME$/webapps (in our example deegree-wcs) you should be able to point to your WCS (OWS) via the following URL:

http://my.server.domain/deegree-wcs/services?

Appendix A Example WCS configuration/capabilities document

Example for a full deegree WCS configuration/capabilities document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WCS_Capabilities xmlns="http://www.opengis.net/wcs"
xmlns:deegree="http://www.deegree.org/wcs"
  xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www.w3.org/1999/xlink"
  updateSequence="1.0.0" version="1.0.0">
  <!--
    except the deegree-section and the Service section all other settings are
    optional and will be set by default if not available. As a result of this a
    user is able to up a WFS with minimaldefinitions as 'DefaultOnlineResource' and
    'DataDirectory'
  -->
  <deegree:deegreeParam>
    <!--
      The DefaultOnlineResource will be used if a required OnlineResource is
      not defined
    -->
    <deegree:DefaultOnlineResource xlink:href="http://localhost:8080/deegree-
    wcs/services" xlink:type="simple" />
    <!-- optional; default = 100 (MB) -->
    <deegree:CacheSize>250</deegree:CacheSize>
    <!-- maximum time for the execution of a request until an exception of time-
    exceed is thrown. optional; default 5 Minutes -->
    <deegree:RequestTimeLimit>10</deegree:RequestTimeLimit>
    <!--
      list of directories to be scanned for coverages to be served by a WCS.
      deegree will look for configuration files in this directories and add the
      corresponding coverages  to the ContentMetadata section if not already
      present.optional; default: $RootDirectory$/WEB-INF/data
    -->
    <!-- Fature not yet implemented but follows very soon -->
    <deegree:DataDirectoryList>
      <deegree:DataDirectory>.</deegree:DataDirectory>
    </deegree:DataDirectoryList>

  </deegree:deegreeParam>
  <Service>
    <!--
      optional; no default
    -->
    <metadataLink about="http://www.deegree.org"
    gml:remoteSchema="http://www.deegree.org"
    metadataType="TC211" xlink:actuate="onLoad"
    xlink:arcrole="http://www.deegree.org"
    xlink:href="http://www.deegree.org" xlink:role="http://www.deegree.org"
    xlink:show="new" xlink:title="deegree WCS metadata" xlink:type="simple" />
    <!--
      optional; no default
    -->
    <description>deegree WCS being OGC WCS 1.0.0 reference
    implementation</description>
    <!--
      mandatory; if missing 'deegreewcs' will be used as default
    -->
    <name>deegree WCS</name>
    <!--
      mandatory; if missing 'deegreewcs' will be used as default
    -->
    <label>deegree WCS</label>
    <!--
      optional; no default
    -->
    <keywords>
```

```xml
      <keyword>deegree</keyword>
      <keyword>WCS</keyword>
      <type codeSpace="http://www.deegree.org">deegree internal</type>
    </keywords>
    <keywords>
      <keyword>reference implemenation</keyword>
      <keyword>WCS</keyword>
      <type codeSpace="http://www.deegree.org">OGC</type>
    </keywords>
    <!--
      optional; no default
    -->
    <responsibleParty>
      <!--
        mandatory; if missing 'deegree' will be used as default
      -->
      <individualName>Andreas Poth</individualName>
      <!--
        optional; no default
      -->
      <organisationName>lat/lon</organisationName>
      <!--
        optional; no default
      -->
      <positionName>technical director</positionName>
      <!--
        optional; no default
        if contactInfo is defined all sub-elements are are also optional
      -->
      <contactInfo>
        <phone>
          <voice>12345678</voice>
          <voice>87654321</voice>
          <facsimile>656454534323</facsimile>
          <facsimile>31243647</facsimile>
        </phone>
        <address>
          <deliveryPoint>Meckenheimer Allee 176</deliveryPoint>
          <deliveryPoint>Bonner Talweg</deliveryPoint>
          <city>Bonn</city>
          <administrativeArea>NRW</administrativeArea>
          <postalCode>53115</postalCode>
          <country>Germany</country>
          <electronicMailAddress>poth@lat-lon.de</electronicMailAddress>
          <electronicMailAddress>info@lat-lon.de</electronicMailAddress>
        </address>
        <onlineResource xlink:actuate="onLoad" xlink:arcrole="http://www.lat-
        lon.de" xlink:href="http://www.lat-lon.de" xlink:role="http://www.lat-
        lon.de" xlink:show="new" xlink:title="lat/lon homepage" xlink:type="simple"
        />
      </contactInfo>
    </responsibleParty>
    <!--
      mandatory; if missing 'NONE' will be used as default
    -->
    <fees codeSpace="http://www.deegree.org">NONE</fees>
    <!--
      mandatory; if missing 'NONE' will be used as default
    -->
    <accessConstraints codeSpace="http://www.deegree.org">NONE</accessConstraints>
    <accessConstraints codeSpace="http://www.deegree.org">SOME</accessConstraints>
  </Service>

  <!--
  The Capability section is mandatory for OGC WCS and for deegree WCS. Please adapt
  the xlink:href="http://yourserver:yourport/deegree-wcs/services?
  -->
  <Capability>
    <Request>
```

```xml
<GetCapabilities>
  <DCPType>
    <HTTP>
      <Get>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Get>
      <Post>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Post>
    </HTTP>
  </DCPType>
</GetCapabilities>

<DescribeCoverage>
  <DCPType>
    <HTTP>
      <Get>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Get>
      <Post>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Post>
    </HTTP>
  </DCPType>
</DescribeCoverage>
<GetCoverage>
  <DCPType>
    <HTTP>
      <Get>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Get>
      <Post>
        <OnlineResource xlink:actuate="onLoad"
        xlink:arcrole="http://www.deegree.org"
        xlink:href="http://localhost:8080/deegree-wcs/services?"
        xlink:role="http://www.deegree.org" xlink:show="new"
        xlink:title="String" xlink:type="simple" />
      </Post>
    </HTTP>
  </DCPType>
</GetCoverage>
</Request>
<!--
  mandatory: application/vnd.ogc.se_xml will be set as default if missing
-->
<Exception>
  <Format>application/vnd.ogc.se_xml</Format>
  <Format>application/deegree_xml</Format>
</Exception>
```

```xml
    <VendorSpecificCapabilities />
  </Capability>
  <!--
  mandatory; if missing it will be created by deegree and filled with
  'CoverageOfferingBrief' descriptions for all coverages that can be found in
  directories listed in 'DataDirectoryList'. If 'ContentMetadata' isn't defined or
  is empty and no coverages that can be found in directories listed in
  'DataDirectoryList' the configuration is invalid because a (deegree) WCS at least
  have to serve one coverage. If 'ContentMetadata' is defined deegree adds all
  coverages found in directories listed in 'DataDirectoryList' that are not defined
  ContentMetadata automaticly. All attributes of 'ContentMetadata' are optional and
  don't have a default
  -->
  <wcs:ContentMetadata xmlns:wcs="http://www.opengis.net/wcs">
    <CoverageOfferingBrief gml:id="ID000001">
      <description></description>
      <name>saltlakecity</name>
      <label />
      <lonLatEnvelope xmlns:wcs="http://www.opengis.net/wcs" srsName="WGS84(DD)">
        <gml:pos dimension="2">-111.89985862992859,40.71940082135575</gml:pos>
        <gml:pos dimension="2">-111.80644038553243,40.82820182642026</gml:pos>
      </lonLatEnvelope>
      <wcs:keywords>
        <wcs:keyword></wcs:keyword>
      </wcs:keywords>
      <deegree:Configuration>
        ../../data/utah/raster/saltlakecity/wcs_saltlakecity_configuration.xml
      </deegree:Configuration>
    </CoverageOfferingBrief>
  </wcs:ContentMetadata>
</WCS_Capabilities>
```

Appendix B   CoverageDescription example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CoverageDescription xmlns="http://www.opengis.net/wcs"
xmlns:deegree="http://www.deegree.org/wcs" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" updateSequence="String" version="1.0.0">
  <CoverageOffering>
    <description xmlns:wcs="http://www.opengis.net/wcs"></description>
    <name>saltlakecity</name>
    <label xmlns:wcs="http://www.opengis.net/wcs">saltlakecity</label>
    <lonLatEnvelope xmlns:wcs="http://www.opengis.net/wcs" srsName="WGS84(DD)">
      <gml:pos dimension="2">-111.8998528924585,40.71941441838457</gml:pos>
      <gml:pos dimension="2">-111.80644636933437,40.828206328381825</gml:pos>
    </lonLatEnvelope>
    <wcs:keywords xmlns:wcs="http://www.opengis.net/wcs">
      <wcs:keyword></wcs:keyword>
    </wcs:keywords>
    <domainSet>
      <spatialDomain xmlns:wcs="http://www.opengis.net/wcs">
        <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#26912">
          <gml:pos dimension="2">424000.5,4507999.5</gml:pos>
          <gml:pos dimension="2">432000.5,4519999.5</gml:pos>
        </gml:Envelope>
      </spatialDomain>
    </domainSet>
    <rangeSet>
      <RangeSet refSys="http://www.deegree.org" refSysLabel="String"
      semantic="http://www.deegree.org">
        <description xmlns:wcs="http://www.opengis.net/wcs"></description>
        <name>saltlakecity</name>
        <label xmlns:wcs="http://www.opengis.net/wcs">saltlakecity</label>
        <nullValues semantic="http://www.deegree.org" type="xs:integer">
          <interval atomic="false" closure="closed"
          semantic="http://www.deegree.org">
            <min>-99</min>
            <max>0</max>
            <res>1</res>
          </interval>
          <singleValue>-9999</singleValue>
        </nullValues>
      </RangeSet>
    </rangeSet>
    <supportedCRSs>
      <requestCRSs xmlns:wcs="http://www.opengis.net/wcs">EPSG:4326
      EPSG:26912</requestCRSs>
      <requestResponseCRSs
      xmlns:wcs="http://www.opengis.net/wcs">EPSG:26912</requestResponseCRSs>
      <nativeCRSs xmlns:wcs="http://www.opengis.net/wcs">EPSG:26912</nativeCRSs>
    </supportedCRSs>
    <supportedFormats xmlns:wcs="http://www.opengis.net/wcs" nativeFormat="jpg">
      <formats>jpeg</formats>
      <formats>GeoTiff</formats>
      <formats>png</formats>
    </supportedFormats>
    <supportedInterpolations default="nearest neighbor">
      <interpolationMethod>nearest neighbor</interpolationMethod>
    </supportedInterpolations>
    <deegree:Extension xmlns:wcs="http://www.opengis.net/wcs" type="shapeIndexed">
      <deegree:Resolution max="99999999" min="16.0">
        <deegree:Range>
          <deegree:Name>default</deegree:Name>
        </deegree:Range>
        <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
        tileProperty="FILENAME">sh16.0</deegree:Shape>
      </deegree:Resolution>
      <deegree:Resolution max="16.0" min="8.0">
        <deegree:Range>
          <deegree:Name>default</deegree:Name>
        </deegree:Range>
```

```xml
        <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
        tileProperty="FILENAME">sh8.0</deegree:Shape>
      </deegree:Resolution>
      <deegree:Resolution max="8.0" min="4.0">
        <deegree:Range>
          <deegree:Name>default</deegree:Name>
        </deegree:Range>
        <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh4.0</deegree:Shape>
      </deegree:Resolution>
      <deegree:Resolution max="4.0" min="2.0">
        <deegree:Range>
          <deegree:Name>default</deegree:Name>
        </deegree:Range>
        <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh2.0</deegree:Shape>
      </deegree:Resolution>
      <deegree:Resolution max="2.0" min="0.0">
        <deegree:Range>
          <deegree:Name>default</deegree:Name>
        </deegree:Range>
        <deegree:Shape directoryProperty="FOLDER" srsName="EPSG:26912"
tileProperty="FILENAME">sh1.0</deegree:Shape>
      </deegree:Resolution>
    </deegree:Extension>
  </CoverageOffering>
</CoverageDescription>
```

Appendix C Deployment Descriptor (web.xml)

Path of file: wms_home/WEB-INF

```xml
<?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>
  <display-name>deegree 2.2</display-name>
  <description>deegree 2.2 OWS</description>
  <servlet>
    <servlet-name>owservice</servlet-name>
    <servlet-class>org.deegree.enterprise.servlet.OGCServletController</servlet-
class>
    <init-param>
      <param-name>services</param-name>
      <param-value>wcs</param-value>
      <description>list of supported services, e.g.: wfs,wms (comma separated)
      allways use lowercase</description>
    </init-param>

    <!-- WMS INITIALIZING PARAMETERS -->
    <!-- <init-param>
      <param-name>wms.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WMSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wms.config</param-name>
      <param-value>WEB-INF/conf/wms/wms_configuration.xml</param-value>
    </init-param>
    -->
    <!-- WFS INITIALIZING PARAMETERS -->
    <!-- <init-param>
      <param-name>wfs.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WFSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wfs.config</param-name>
      <param-value>WEB-INF/conf/wms/LOCALWFS_capabilities.xml</param-value>
    </init-param>
    -->
    <!-- WCS INITIALIZING PARAMETERS -->

    <init-param>
      <param-name>wcs.handler</param-name>
      <param-value>org.deegree.enterprise.servlet.WCSHandler</param-value>
    </init-param>
    <init-param>
      <param-name>wcs.config</param-name>
      <param-value>WEB-INF/conf/wcs/wcs_configuration.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>owservice</servlet-name>
    <url-pattern>/services</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>/index.jsp</welcome-file>
  </welcome-file-list>

  <error-page>
    <error-code>500</error-code>
    <location>/error.jsp</location>
  </error-page>
  <error-page>
```

```xml
<?xml version="1.0"?>
```

```xml
    <exception-type>org.deegree.ogcwebservices.OGCWebServiceException</exception-
    type>
    <location>/error.jsp</location>
  </error-page>
</web-app>
```