D3.2 Big Data Toolbox Training Manual I

WP3 – Large Scale Demonstrators

Authors: Dimitar Misev, Peter Baumann, Gedas Vaitkus

Date: 11.02.20

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824478.

Full Title	**Promoting the international competitiveness of European Remote Sensing companies through cross-cluster collaboration**		
Grant Agreement No	824478	**Acronym**	PARSEC
Start date	1st May 2019	**Duration**	30 months
EU Project Officer	Milena Stoyanova		
Project Coordinator	Emmanuel Pajot (EARSC)		
Date of Delivery	**Contractual** 29.02.2020	**Actual**	26.02.2020
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	RASDAMAN		
Lead Author	Dimitar Misev	**Email**	misev@rasdaman.com
Other authors	Peter Baumann (RASDAMAN), Gedas Vaitkus (GEOMATRIX)		
Reviewer(s)	Weronika Borejko (EARSC)		
Keywords	big data, EO, datacubes, array databases, OGC services, rasdaman, sagris		

Document History

Version	Issue date	Stage	Changes	Contributor
1.0	11.02.2020	Draft	First draft	RASDAMAN
1.1	12.02.2020	Draft	Add list of acronyms	RASDAMAN

Table of Contents

Table of Figures

Table of Tables

List of Acronyms

API	Application Programming Interface
CRS	Coordinate Reference System
EO	Earth Observation
OGC	Open Geospatial Consortium
WMS	Web Map Service
WCPS	Web Coverage Processing Service
WCS	Web Coverage Service

Executive Summary

This document is a companion training manual for the Big Data Toolbox service offered by PARSEC. The public user-facing interfaces and API of the Big Data Toolbox are comprised of standard OGC services for big EO datacubes:

- WCS for downloading datacubes in desired projection and format, with flexible spatio-temporal and range subsetting applied as needed;
- WCPS for doing filtering, processing, and analytics on datacubes through a powerful but concise and safe declarative query language;
- WMS for visualizing and exploring datacubes, usually as maps in the browser.

The training manual aims to be pragmatic in style and focuses on serving as a concise introduction to these standard interfaces, with simple practical examples to aid quick understanding. As the Big Data Toolbox service is powered by a rasdaman server on the backend, the documentation and mailing lists of rasdaman can be considered as additional resources for more advanced topics not explicitly covered in this document. Additionally, the standard documents published by OGC are useful as canonical references.

1. Datacube Download with WCS

The OGC Web Coverage Service (WCS) standard defines support for modeling and retrieval of geospatial data as *coverages* (e.g. sensor, image, or statistics data).

WCS consists of a *Core* specification for basic operation support with regards to coverage discovery and retrieval, and various *Extension* specifications for optional capabilities that a service could provide on offered coverage objects.

1.1 Core

The Core specification is agnostic of implementation details, hence, access syntax and mechanics are defined by *protocol extensions*: KVP/GET, XML/POST, and XML/SOAP. Rasdaman supports all three, but further on the examples are in *KVP/GET* exclusively, as it is the most straightforward way for constructing requests by appending a standard query string to the service endpoint URL. Commonly, for all operations the KVP/GET request will look as follows:

```
http(s)://<endpoint url>?service=WCS
                        &version=2.0.1
                        &request=<operation>
                        &...
```

Three fundamental operations are defined by the Core:

- **GetCapabilities** - returns overal service information and a list of available coverages; the request looks generally as above, with the <*operation*> being GetCapabilities:

  ```
  http(s)://<endpoint url>?service=WCS&version=2.0.1
                          &request=GetCapabilities
  ```

  Example:
  http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCapabilities

- **DescribeCoverage** - detailed description of a specific coverage:

  ```
  http(s)://<endpoint url>?service=WCS&version=2.0.1
                          &request=DescribeCoverage
                          &coverageId=<coverage id>
  ```

  Example:
  http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=DescribeCoverage&coverageId=AvgLandTemp

- **GetCoverage** - retreive a whole coverage, or arbitrarily restricted on any of its axes whether by new lower/upper bounds (*trimming*) or at a single index (*slicing*):

```
http(s)://<endpoint url>?service=WCS&version=2.0.1
                        &request=GetCoverage
                        &coverageId=<coverage id>
        [optional]      &subset=<axis>(<lower>:<upper>)
        [optional]      &subset=<axis>(<index>)
        [optional]      &format=<mime type>
```

Example which reduces axis Lon and slices on the ansi time axis:
http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=AvgLandTemp&subset=Lon(-90.0,85.3)&subset=ansi(“2014-10-01”)&format=image/jpeg

1.2 Updating

The Transaction extension (WCS-T) specifies the following operations for constructing, maintenance, and removal of coverages on a server: *InsertCoverage*, *UpdateCoverage*, and *DeleteCoverage*.

Rasdaman provides the wcst import tool to simplify the ingestion of data into analysis-ready coverages (aka datacubes) by generating WCS-T requests as instructed by a simple configuration file.

1.3 Processing

The Processing extension enables advanced analytics on coverages through WCPS queries. The request format is as follows:

```
http(s)://<endpoint url>?service=WCS&version=2.0.1
                        &request=ProcessCoverages
                        &query=<wcps query>
```

E.g. calculate the average on the subset from the previous GetCoverage example:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=ProcessCoverages&query=for $c in (AvgLandTemp) return avg($c[Lon(-90.0:85.3), ansi(“2014-10-01”)])

1.4 Range subsetting

The cell values of some coverages consist of multiple components (also known as ranges, bands, channels, fields, attributes). The Range subsetting extension specifies the extraction and/or recombination in possibly different order of one or more bands. This is done by listing the wanted bands or band intervals; e.g *AverageChlorophyllScaled* has Blue, Green, and Red bands and the following recombines them into a Red, Green, Blue order:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=AverageChlorophyllScaled&format=image/png&subset=unix(“2015-01-01”)&rangesubset=Red,Green,Blue

1.5 Scaling

Scaling up or down is a common operation supported by the Scaling extension. An additional GetCoverage parameter indicates the scale factor in several possible ways: as a single number applying to all axes, multiple numbers applying to individual axes, full target scale domain, or per-axis target scale domains. E.g. a single factor to downscale all axes by 4x:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=AvgLandTemp&subset=ansi(“2014-10-01”)&format=image/jpeg&scaleFactor=0.25

1.6 Reprojection

The CRS extension allows to reproject a coverage before retreiving it. For example `AverageChlorophyllScaled` has native CRS EPSG:4326, and the following request will return the result in EPSG:3857:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=AverageChlorophyllScaled&format=image/png&subset=unix(“2015-01-01”)&outputCrs=http://ows.rasdaman.org/def/crs/EPSG/0/3857

or change the CRS in which subset or scale coordinates are specified:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=AverageChlorophyllScaled&format=image/png&subset=Lon(0,10000000)&subset=Lat(0,20000000)&subset=unix(%222015-01-01%22)&subsettingCrs=http://ows.rasdaman.org/def/crs/EPSG/0/3857

1.7 Interpolation

Scaling or reprojection can be performed with various interpolation methods as enabled by the Interpolation extension:

http://ows.rasdaman.org/rasdaman/ows?service=WCS&version=2.0.1&request=GetCoverage&coverageId=mean_summer_airtemp&outputCrs=http://ows.rasdaman.org/def/crs/EPSG/0/3857&interpolation=http://www.opengis.net/def/interpolation/OGC/1/cubic

Rasdaman supports several interpolations as documented here.

2. Datacube Analytics with WCPS

The OGC Web Coverage Processing Service (WCPS) standard defines a protocol-independent declarative query language for the extraction, processing, and analysis of multi-dimensional coverages representing sensor, image, or statistics data.

The overall execution model of WCPS queries is similar to XQuery FLOWR:

```
for $covIter1 in (covName, ...),
    $covIter2 in (covName, ...),
    ...
let $aliasVar1 := covExpr,
    $aliasVar2 := covExpr,
    ...
where booleanExpr
return processingExpr
```

Any coverage listed in the WCS *GetCapabilities* response can be used in place of covName. Multiple $covIter essentially translate to nested loops. For each iteration, the return clause is evaluated if the result of the where clause is true. Coverage iterators and alias variables can be freely used in where / return expressions.

Conforming WCPS queries can be submitted to rasdaman as WCS ProcessCoverages requests, e.g:

```
http://localhost:8080/rasdaman/ows?service=WCS&version=2.0.1
    &request=ProcessCoverages
    &query=for $covIter in (covName) ...
```

The *WCS-client* deployed with every rasdaman installation provides a convenient console for interactively writing and executing WCPS queries: open http://localhost:8080/rasdaman/ows in your Web browser and proceed to the *ProcessCoverages* tab.

Operations can be categorized by the type of data they result in: scalar, coverage, or metadata.

1.8 Scalar operations

- **Standard operations** applied on scalar operands return scalar results:

Operation category	Operations
Arithmetic	`+ - * / abs round`
Exponential	`exp log ln pow sqrt`

Trigonometric	`sin cos tan sinh cosh tanh`
	`arcsin arccos arctan`

Comparison	`> < >= <= = !=`
Logical	`and or xor not bit overlay`
Select field from multiband value	`.`
Create multiband value	`{ bandName: value; ..., bandName: value }`
Type casting	`(baseType) value`

	where baseType is one of: boolean, [unsigned] char / short / int / long, float, double, cint16, cint32, cfloat32, cfloat64

Table 1. Standard operations returning scalar values.

- **Aggregation operations** summarize coverages into a scalar value.

Aggregation type	Function / Expression
Of numeric coverages	`avg`, `add`, `min`, `max`
Of boolean coverages	`count` number of true values; `some`/`all` = true if some/all values are true
General condenser	`condense` *op* `over` $iterVar axis(lo:hi), … [`where` boolScalarExpr] `using` scalarExpr

Table 2. Aggregation operations.

The *general condenser* aggregates values across an iteration domain with a condenser operation *op* (one of +, *, max, min, and, or or). For each coordinate in the iteration domain defined by the over clause, the scalar expression in the using clause is evaluated and added to the final aggregated result; the optional where clause allows to filter values from the aggregation.

1.9 Coverage operations

- **Standard operations** applied on coverage (or mixed coverage and scalar) operands return coverage results. The operation is applied pair-wise on each cell from the coverage operands, or on the scalars and each cell from the coverage in case some operands are scalars. All coverage operands must have matching domains and CRS.

- **Subsetting** allows to select a part of a coverage (or crop it to a smaller domain):

```
covExpr[ axis1(lo:hi), axis2(slice), axis3:crs(...), ... ]
```

1. `axis1` in the result is reduced to span from coordinate `lo` to `hi`. Either or both `lo` and `hi` can be indicated as *, corresponding to the minimum or maximum bound of that axis.
2. `axis2` is restricted to the exact slice coordinate and removed from the result.
3. `axis3` is subsetted in coordinates specified in the given `crs`. By default coordinates must be given in the native CRS of C.

- **Extend** is similar to subsetting but can be used to enlarge a coverage with null values as well, i.e. lo and hi can extend beyond the min/max bounds of a particular axis; only trimming is possible:

```
extend( covExpr, { axis1(lo:hi), axis2:crs(lo:hi), ... } )
```

- **Scale** is like extend but it resamples the current coverage values to fit the new domain:

  ```
  scale( covExpr, { axis1(lo:hi), axis2:crs(lo:hi), ... } )
  ```

- **Reproject** allows to change the CRS of the coverage:

  ```
  crsTransform( covExpr, { axis1:crs1, axis2:crs2, ... } )
  ```

- **Conditional evaluation** is possible with the switch statement:

  ```
  switch
    case boolCovExpr return covExpr
    case boolCovExpr return covExpr
    ...
    default return covExpr
  ```

- **General coverage constructor** allows to create a coverage given a domain, where for each coordinate in the domain the value is dynamically calculated from a value expression which potentially references the iterator variables:

  ```
  coverage covName
  over $iterVar axis(lo:hi), ...
  values scalarExpr
  ```

- **General condenser on coverages** is same as the scalar general condenser, except that in the using clause we have a coverage expression. The coverage values produced in each iteration are cell-wise aggregated into a single result coverage.

  ```
  condense op
  over $iterVar axis(lo:hi), ...
  [ where boolScalarExpr ]
  values covExpr
  ```

- **Encode** allows to export coverages in a specified data format, e.g:

  ```
  encode(covExpr, "image/jpeg")
  ```

1.10 Metadata operations

Several functions allow to extract metadata information about a coverage C:

Metadata function	Result
imageCrsDomain(C, a)	Grid (lo, hi) bounds for axis a.
domain(C, a, c)	Geo (lo, hi) bounds for axis a in CRS c.
crsSet(C)	Set of CRS identifiers.
nullSet(C)	Set of null values.

Table 3. Metadata operations.

3. Datacube Portrayal with WMS

The OGC Web Map Service (WMS) standard defines map portrayal on geo-spatial data. In rasdaman, a WMS service can be enabled on any coverage, including 3-D or higher dimensional; the latest 1.3.0 version is supported.

rasdaman supports two operations: *GetCapabilities*, *GetMap* from the standard. We will not go into the details, as users do not normally hand-write WMS requests but let a client tool or library generate them instead. Please check the Clients section for some examples.

4. Clients

1.11 Rasdaman WSClient

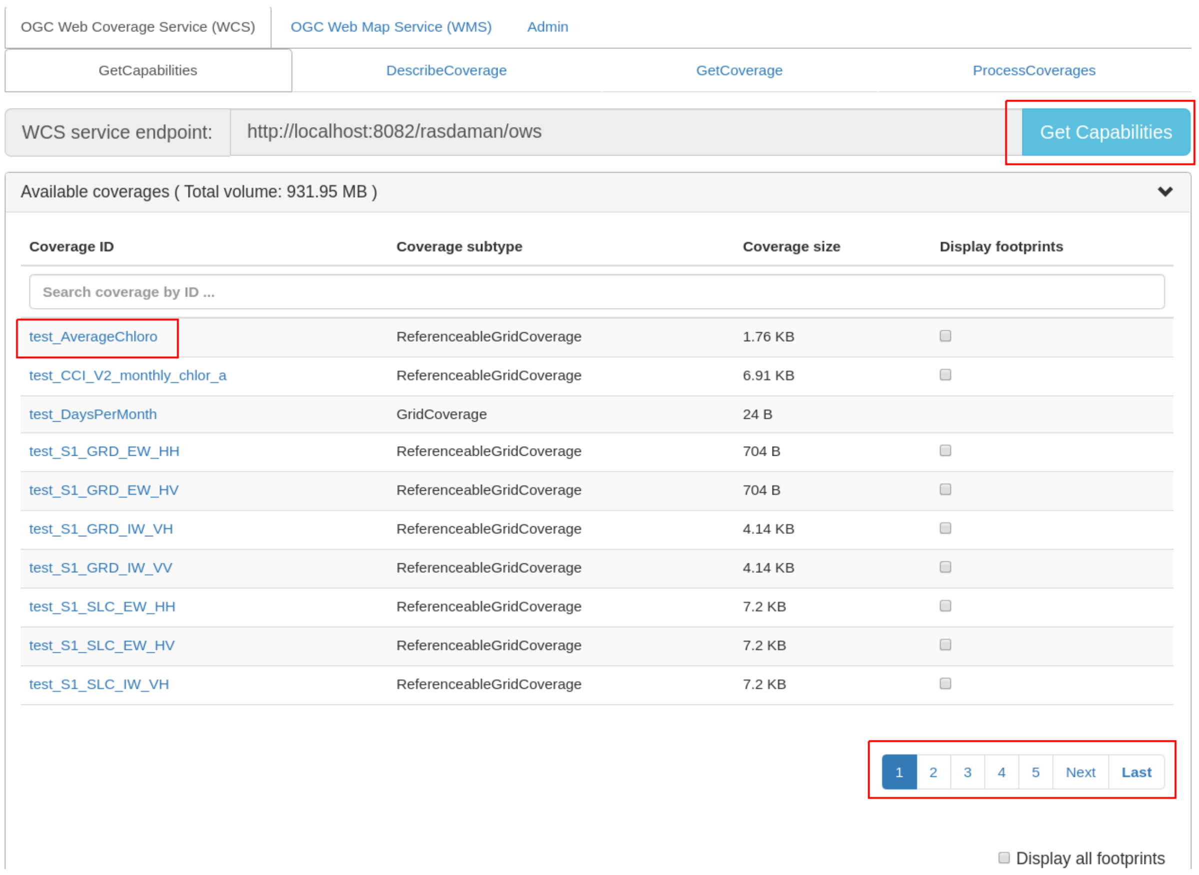
WSClient is a web-client application to interact with WCS (version 2.0.1) and WMS (version 1.3.0) compliant servers. Once rasdaman is installed it is usually accessible at `http://localhost:8080/rasdaman/ows`; a publicly accessible example is available at http://ows.rasdaman.org/rasdaman/ows. The client has three main tabs: `OGC Web Coverage Service (WCS)`, `OGC Web Map Service (WMS)` and `Admin`. Further on, the functionality in each tab is described in details.

1.11.1 WCS

There are sub-tabs for each of OGC WCS standard requests: GetCapabilities, DescribeCoverage, GetCoverage, ProcessCoverages.

GetCapabilities

This is the default tab when accessing the WSClient. It lists all coverages available at the specified WCS endpoint. Clicking on the `Get Capabilities` button will reload the coverages list. One can also search a coverage by typing the first characters of its name in the text box. Clicking on a coverage name will move to `DescribeCoverage` tab to view its metadata.



WCS service endpoint: http://localhost:8082/rasdaman/ows

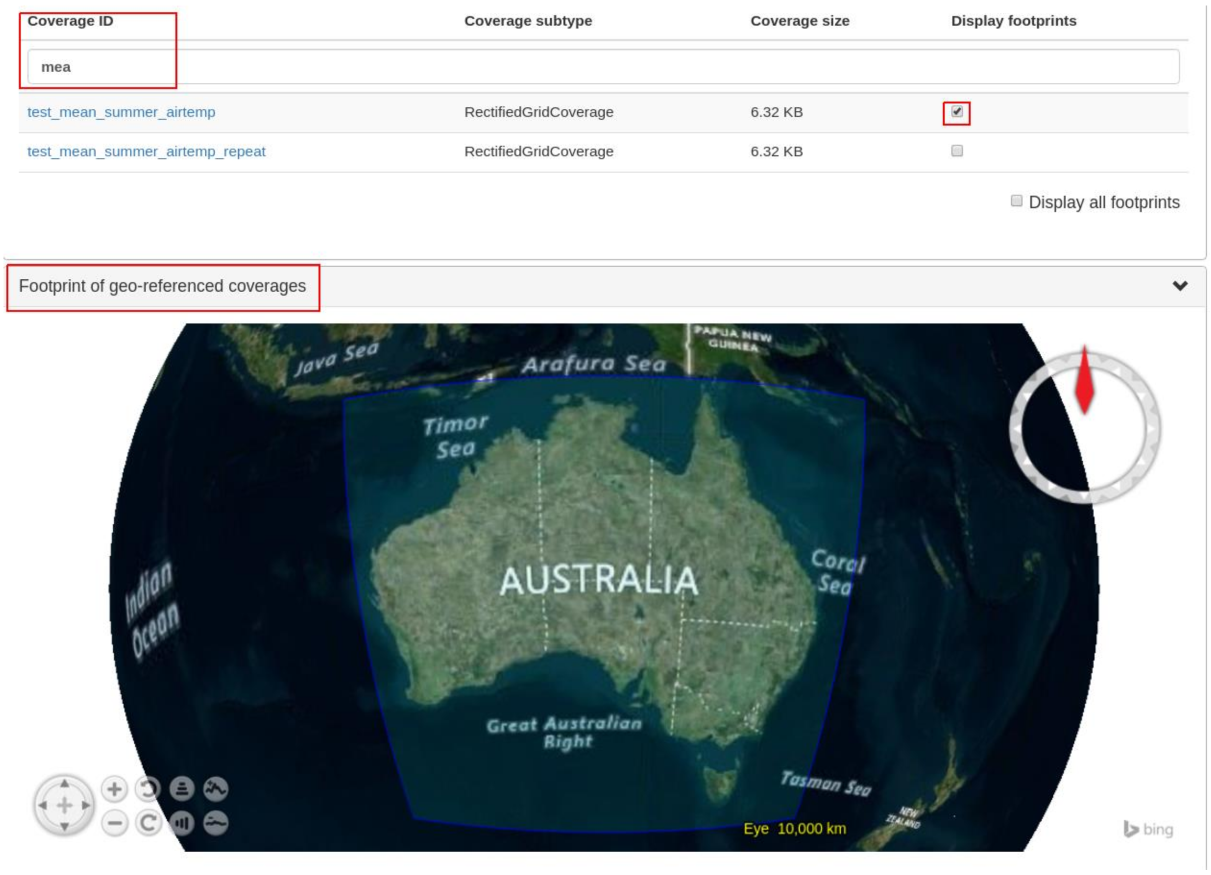
Available coverages (Total volume: 931.95 MB)

Search coverage by ID ...

Coverage ID	Coverage subtype	Coverage size	Display footprints
test_AverageChloro	ReferenceableGridCoverage	1.76 KB	☐
test_CCI_V2_monthly_chlor_a	ReferenceableGridCoverage	6.91 KB	☐
test_DaysPerMonth	GridCoverage	24 B	
test_S1_GRD_EW_HH	ReferenceableGridCoverage	704 B	☐
test_S1_GRD_EW_HV	ReferenceableGridCoverage	704 B	☐
test_S1_GRD_IW_VH	ReferenceableGridCoverage	4.14 KB	☐
test_S1_GRD_IW_VV	ReferenceableGridCoverage	4.14 KB	☐
test_S1_SLC_EW_HH	ReferenceableGridCoverage	7.2 KB	☐
test_S1_SLC_EW_HV	ReferenceableGridCoverage	7.2 KB	☐
test_S1_SLC_IW_VH	ReferenceableGridCoverage	7.2 KB	☐

Figure 1 List of coverages shown on the GetCapabilities tab.

If a coverage is geo-referenced, a checkbox will be visible in the `Display footprints` column, allowing to view the coverage's geo bounding box (in EPSG:4326) on the globe below.



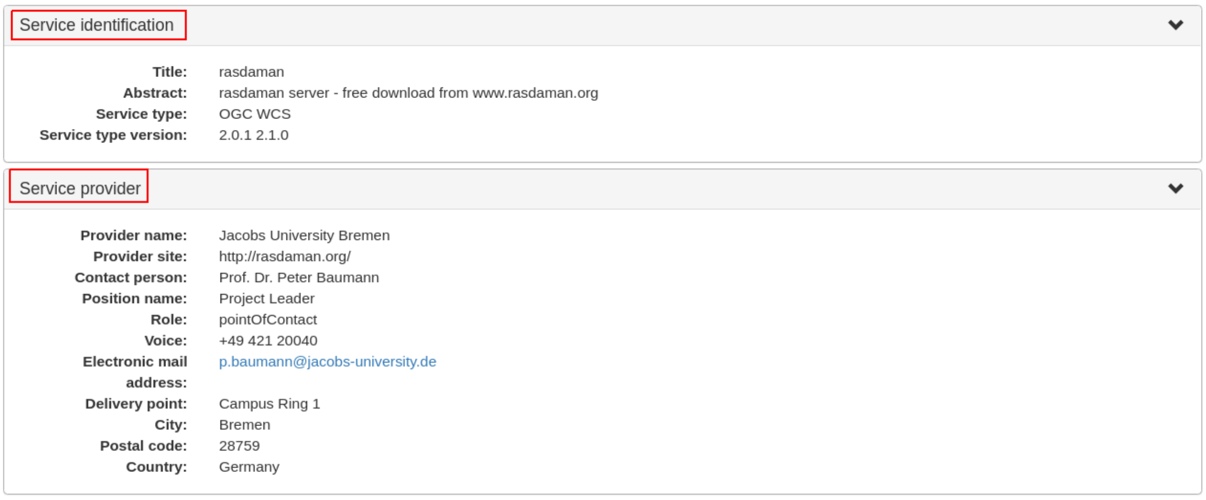
Coverage ID	Coverage subtype	Coverage size	Display footprints
mea			
test_mean_summer_airtemp	RectifiedGridCoverage	6.32 KB	✓
test_mean_summer_airtemp_repeat	RectifiedGridCoverage	6.32 KB	☐

☐ Display all footprints

Footprint of geo-referenced coverages

Figure 2 Selected coverage footprints shown on a globe.

At the bottom the metadata of the OGC WCS service endpoint are shown. These metadata can be changed in the `Admin -> OWS Metadata Management` tab. Once updated in the admin tab, click on `Get Capabilities` button to see the new metadata.



Service identification

Title:	rasdaman
Abstract:	rasdaman server - free download from www.rasdaman.org
Service type:	OGC WCS
Service type version:	2.0.1 2.1.0

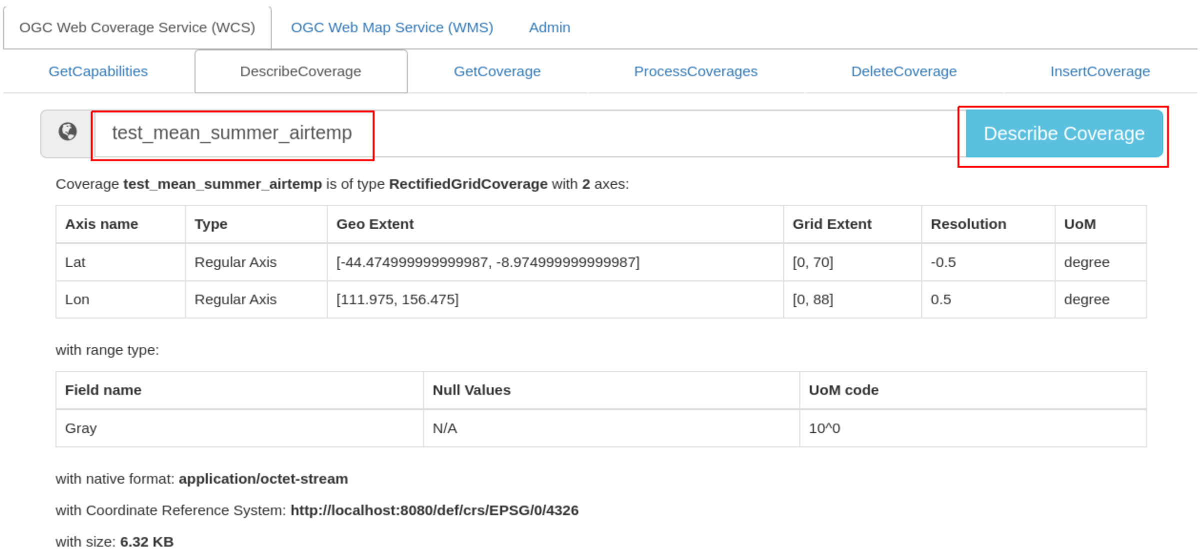
Service provider

Provider name:	Jacobs University Bremen
Provider site:	http://rasdaman.org/
Contact person:	Prof. Dr. Peter Baumann
Position name:	Project Leader
Role:	pointOfContact
Voice:	+49 421 20040
Electronic mail address:	p.baumann@jacobs-university.de
Delivery point:	Campus Ring 1
City:	Bremen
Postal code:	28759
Country:	Germany

Figure 3 WCS service metadata.

DescribeCoverage

Here the full description of a selected coverage can be seen. One can type the first few characters to search for a coverage id and click on Describe Coverage button to view its OGC WCS metadata.



Coverage **test_mean_summer_airtemp** is of type **RectifiedGridCoverage** with **2** axes:

Axis name	Type	Geo Extent	Grid Extent	Resolution	UoM
Lat	Regular Axis	[-44.474999999999987, -8.974999999999987]	[0, 70]	-0.5	degree
Lon	Regular Axis	[111.975, 156.475]	[0, 88]	0.5	degree

with range type:

Field name	Null Values	UoM code
Gray	N/A	10^0

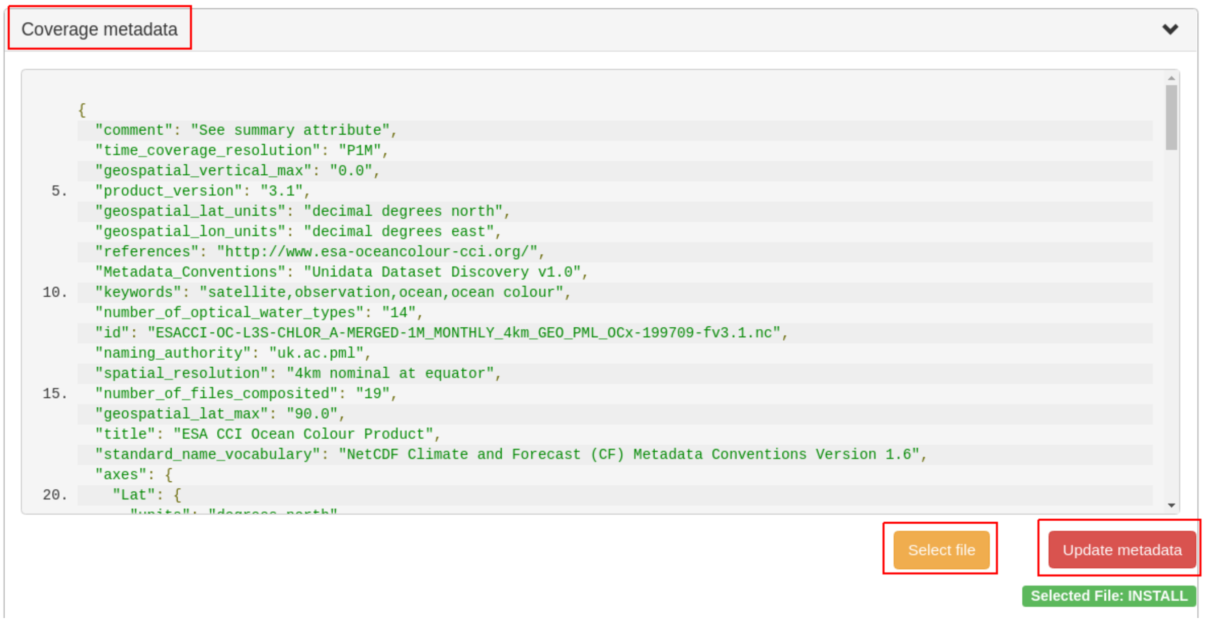
with native format: **application/octet-stream**

with Coordinate Reference System: **http://localhost:8080/def/crs/EPSG/0/4326**

with size: **6.32 KB**

Figure 4 Showing full description of a coverage.

Once logged in as admin, it's possible to replace the metadata with one from a valid XML or JSON file.

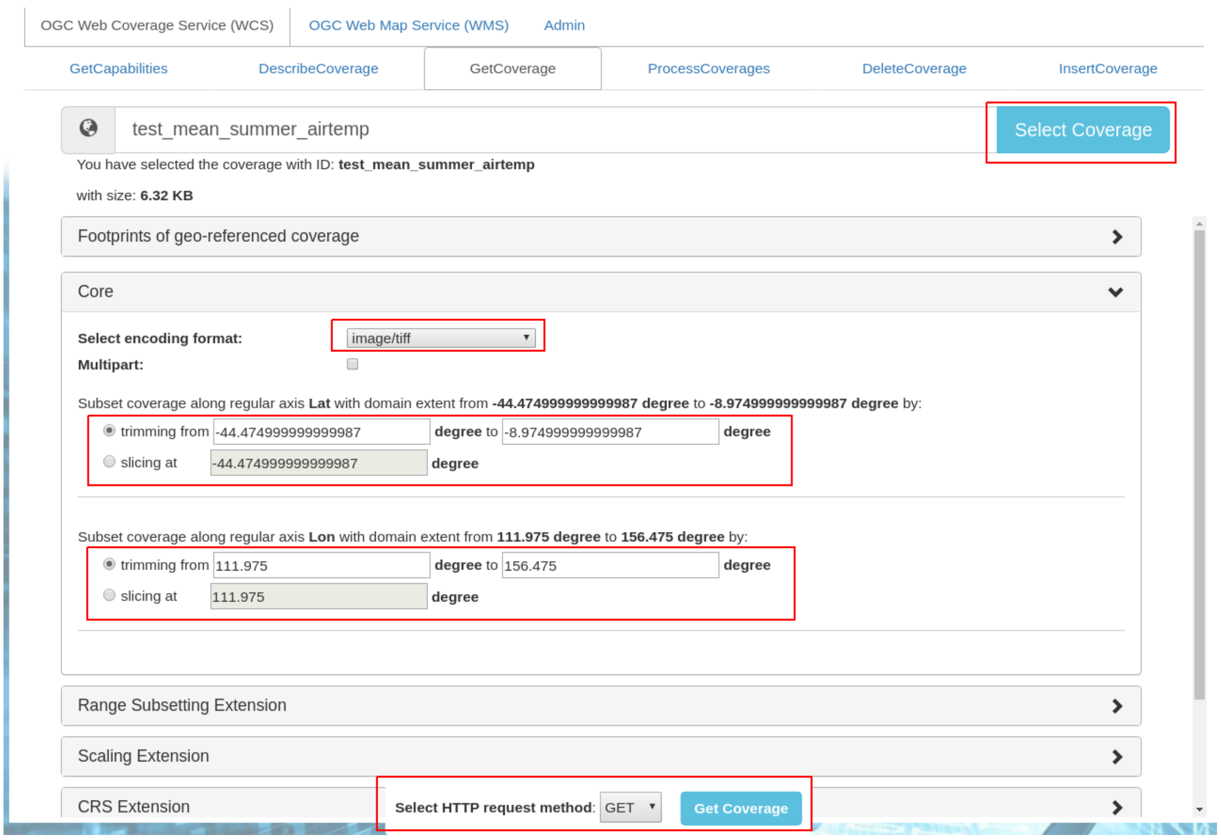


```
{
  "comment": "See summary attribute",
  "time_coverage_resolution": "P1M",
  "geospatial_vertical_max": "0.0",
  "product_version": "3.1",
  "geospatial_lat_units": "decimal degrees north",
  "geospatial_lon_units": "decimal degrees east",
  "references": "http://www.esa-oceancolour-cci.org/",
  "Metadata_Conventions": "Unidata Dataset Discovery v1.0",
  "keywords": "satellite,observation,ocean,ocean colour",
  "number_of_optical_water_types": "14",
  "id": "ESACCI-OC-L3S-CHLOR_A-MERGED-1M_MONTHLY_4km_GEO_PML_OCx-199709-fv3.1.nc",
  "naming_authority": "uk.ac.pml",
  "spatial_resolution": "4km nominal at equator",
  "number_of_files_composited": "19",
  "geospatial_lat_max": "90.0",
  "title": "ESA CCI Ocean Colour Product",
  "standard_name_vocabulary": "NetCDF Climate and Forecast (CF) Metadata Conventions Version 1.6",
  "axes": {
    "Lat": {
```

Figure 5 Updating the metadata of a coverage.

GetCoverage

Downloading coverage *data* can be done on this tab (or the next one, ProcessCoverages). It's similiarly possible search for a coverage id in the text box and click on `Select Coverage` button to view its boundaries. Depending on the coverage dimension, one can do trim or slice subsets on the corresponding axes to select an area of interest. The output format can be selected (provided it supports the output dimension). Finally, clicking on `Get Coverage` button will download the coverage.



test_mean_summer_airtemp

You have selected the coverage with ID: **test_mean_summer_airtemp**

with size: **6.32 KB**

Select encoding format: image/tiff
Multipart: ☐

Subset coverage along regular axis **Lat** with domain extent from **-44.474999999999987 degree** to **-8.974999999999987 degree** by:

● trimming from -44.474999999999987 **degree** to -8.974999999999987 **degree**
○ slicing at -44.474999999999987 **degree**

Subset coverage along regular axis **Lon** with domain extent from **111.975 degree** to **156.475 degree** by:

● trimming from 111.975 **degree** to 156.475 **degree**
○ slicing at 111.975 **degree**

Select HTTP request method: GET

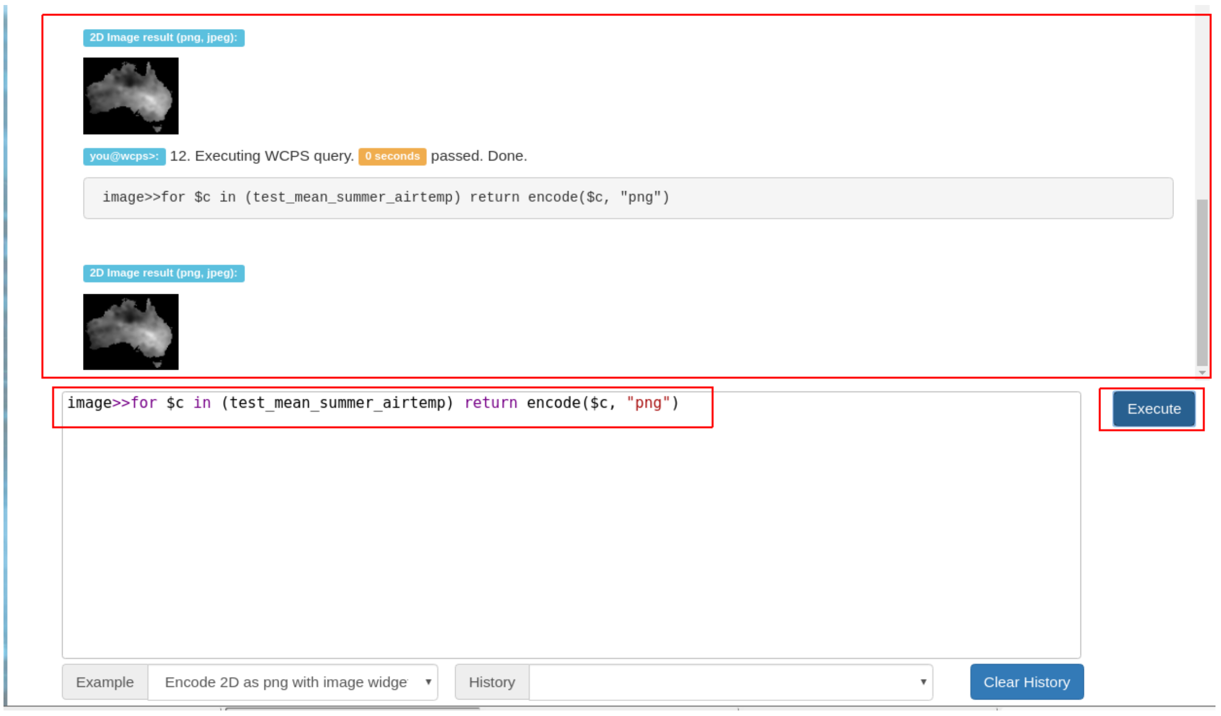
Figure 6 Downloading a subset of a coverage, encoded in image/tiff.

In addition, further parameters can be specified as supported by the WCS extensions, e.g. scaling factor, output CRS, subset of ranges (bands), etc.

ProcessCoverages

WCPS queries can be typed in a text box. Once `Excute` is clicked, the result will be

- displayed on the output console if it's a scalar or the query was prefixed with `image>>` (for 2D png/jpeg) or `diagram>>` for (1D csv/json);
- otherwise it will be downloaded.


2D Image result (png, jpeg):

you@wcps>: 12. Executing WCPS query. 0 seconds passed. Done.

```
image>>for $c in (test_mean_summer_airtemp) return encode($c, "png")
```

2D Image result (png, jpeg):

```
image>>for $c in (test_mean_summer_airtemp) return encode($c, "png")
```

Figure 7 Query and output areas on the ProcessCoverages tab.

DeleteCoverage

This tab allows to *delete* a specific coverage from the server. It is only visible when logged in the `Admin` tab.

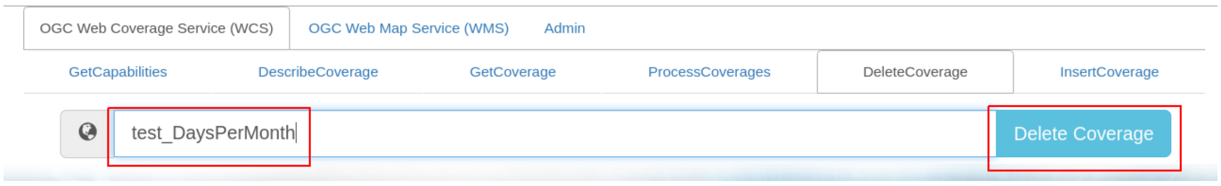


Figure 8 Deleting coverage test_DaysPerMonth.

InsertCoverage

Similarly, this tab is only visible when logged in the Admin tab. To insert a coverage, a URL pointing to a valid coverage definition according to the WCS-T standard needs to be provided. Clicking on Insert Coverage button will invoke the correct WCS-T request on the server.

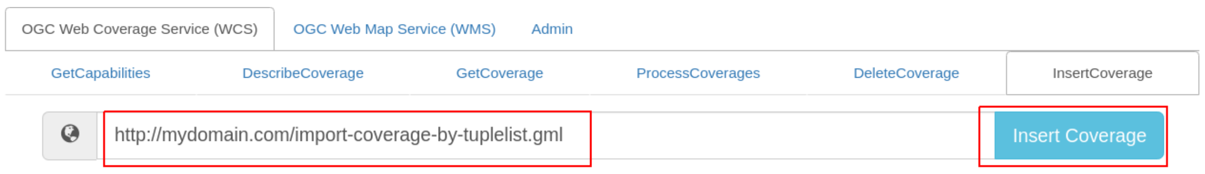

OGC Web Coverage Service (WCS)
OGC Web Map Service (WMS)
Admin
GetCapabilities
DescribeCoverage
GetCoverage
ProcessCoverages
DeleteCoverage
InsertCoverage
http://mydomain.com/import-coverage-by-tuplelist.gml
Insert Coverage

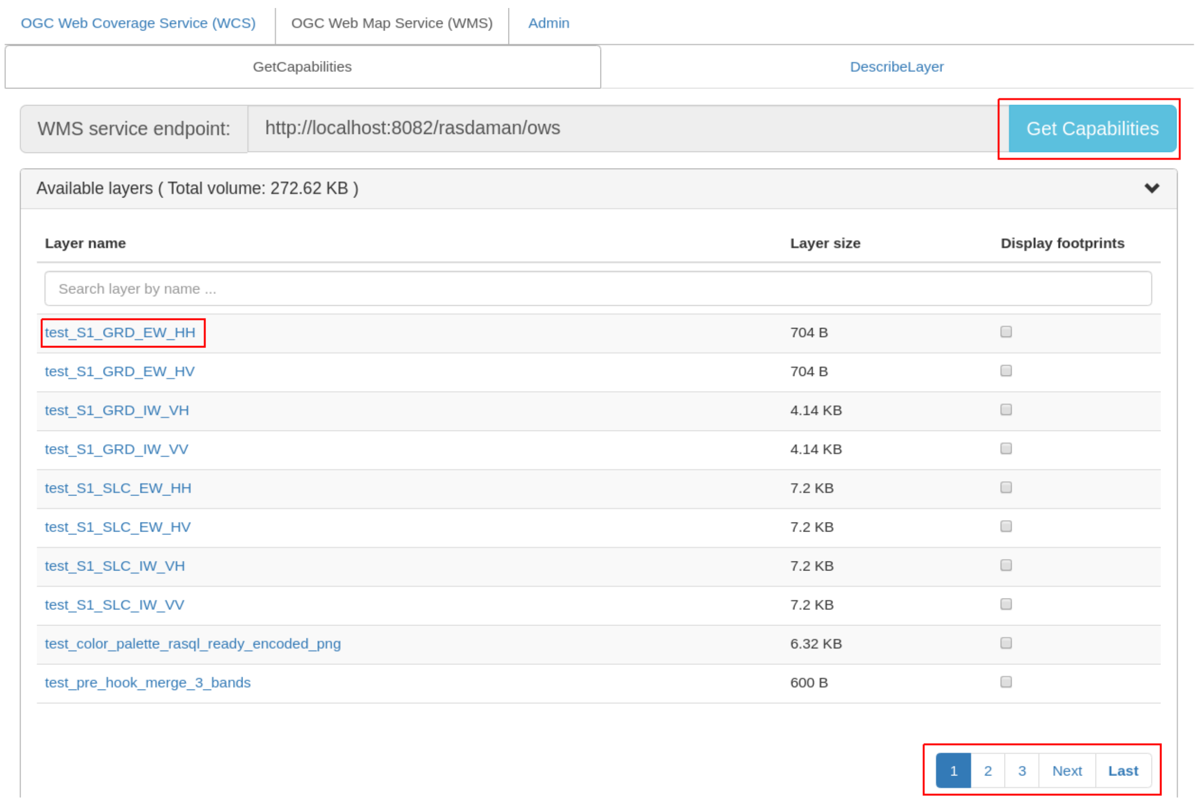
Figure 9 Inserting a coverage given a URL pointing to a valid GML document.

1.11.2 WMS

This tab contain sub-tabs which are related to the supported OGC WMS requests.

GetCapabilities

This tab lists the available layers on the specified server. To reload the list, click on the `Get Capabilities` button. Clicking on a layer name will move to `DescribeLayer` tab to view its description.

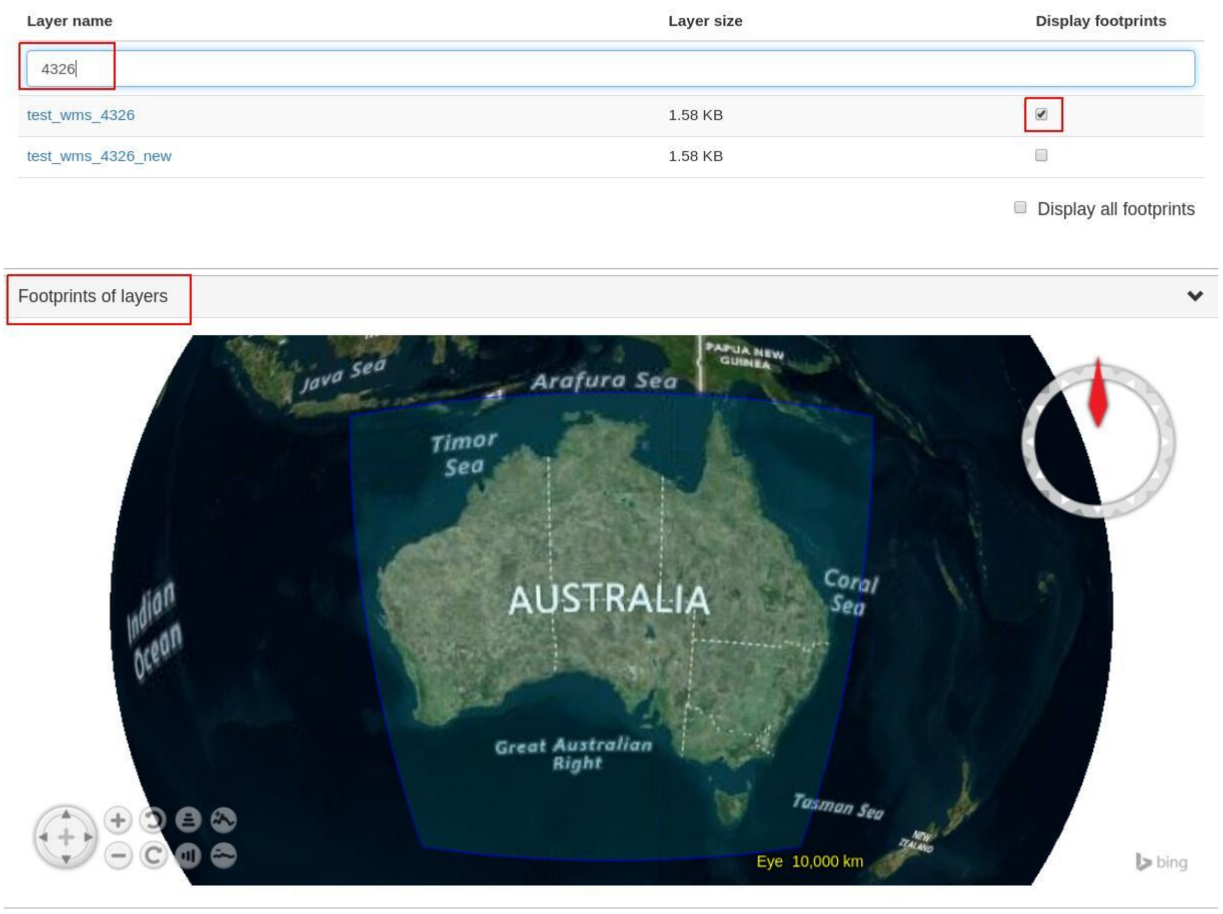


Available layers (Total volume: 272.62 KB)

Layer name	Layer size	Display footprints
test_S1_GRD_EW_HH	704 B	☐
test_S1_GRD_EW_HV	704 B	☐
test_S1_GRD_IW_VH	4.14 KB	☐
test_S1_GRD_IW_VV	4.14 KB	☐
test_S1_SLC_EW_HH	7.2 KB	☐
test_S1_SLC_EW_HV	7.2 KB	☐
test_S1_SLC_IW_VH	7.2 KB	☐
test_S1_SLC_IW_VV	7.2 KB	☐
test_color_palette_rasql_ready_encoded_png	6.32 KB	☐
test_pre_hook_merge_3_bands	600 B	☐

Figure 10 List of layers shown on the GetCapabilities tab.

Similar to the WCS GetCapabilities tab, it's possible to search for layer names, or show their footprints.



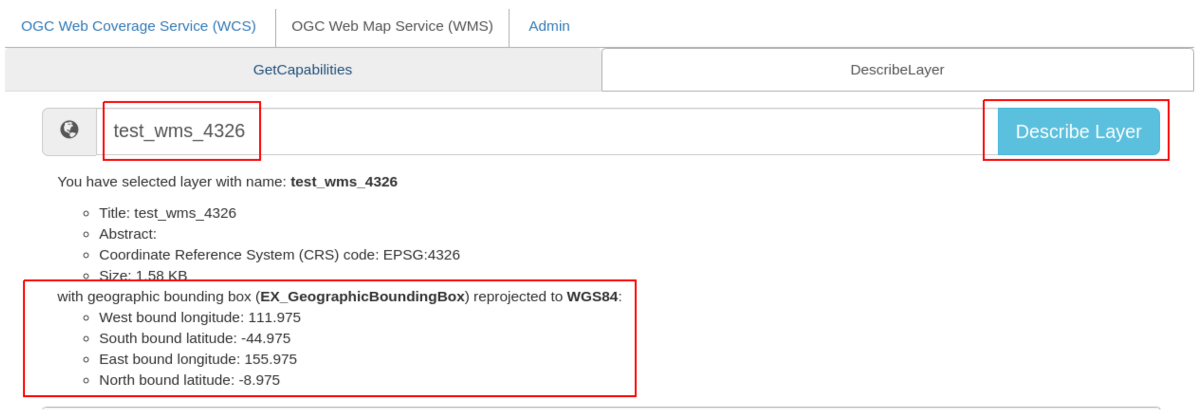
Layer name	Layer size	Display footprints
4326		
test_wms_4326	1.58 KB	☑
test_wms_4326_new	1.58 KB	☐

☐ Display all footprints

Figure 11 Selected layer footprints shown on a globe.

DescribeLayer

Here the full description of a selected layer is shown. One can type the first few characters to search for a layer name and click on `Describe Layer` button to view its OGC WMS metadata.



You have selected layer with name: **test_wms_4326**

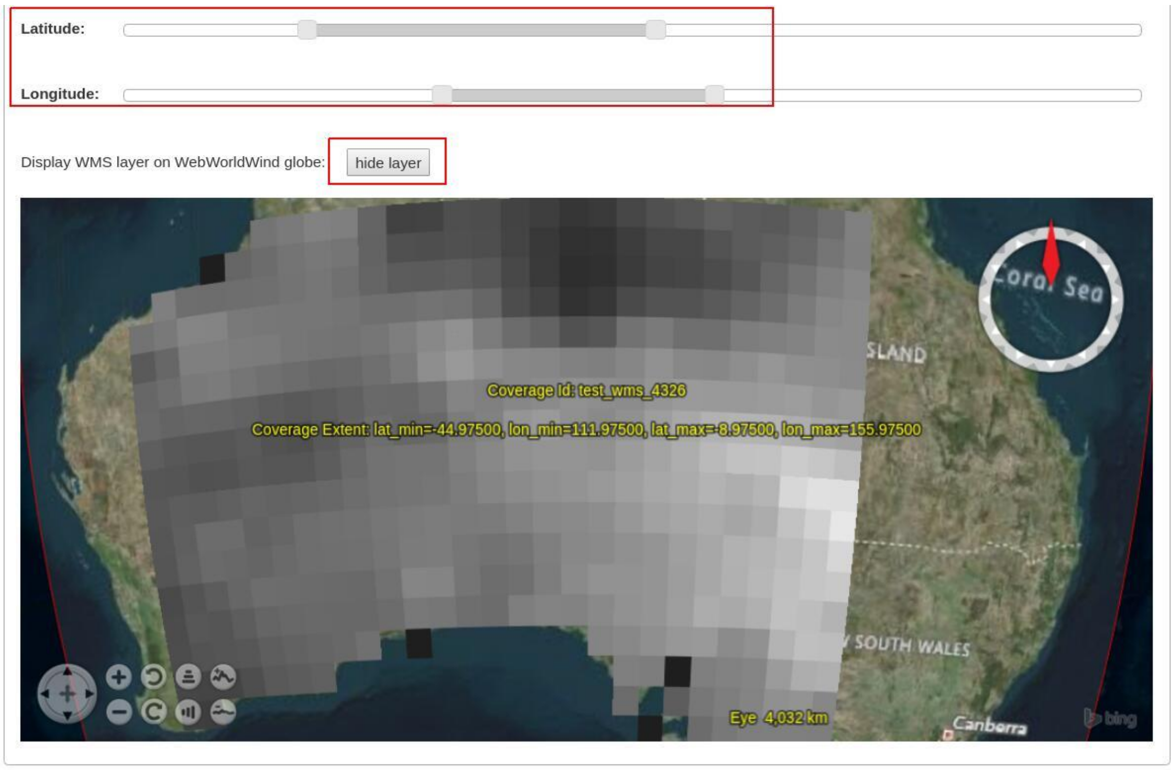
- Title: test_wms_4326
- Abstract:
- Coordinate Reference System (CRS) code: EPSG:4326
- Size: 1.58 KB

with geographic bounding box (**EX_GeographicBoundingBox**) reprojected to **WGS84**:

- West bound longitude: 111.975
- South bound latitude: -44.975
- East bound longitude: 155.975
- North bound latitude: -8.975

Figure 12 Showing full description of a layer.

Depending on layer's dimension, one can click on show layer button and interact with axes' sliders to view a layer's slice on the globe below. Click on the hide layer button to hide the displayed layer on the globe.

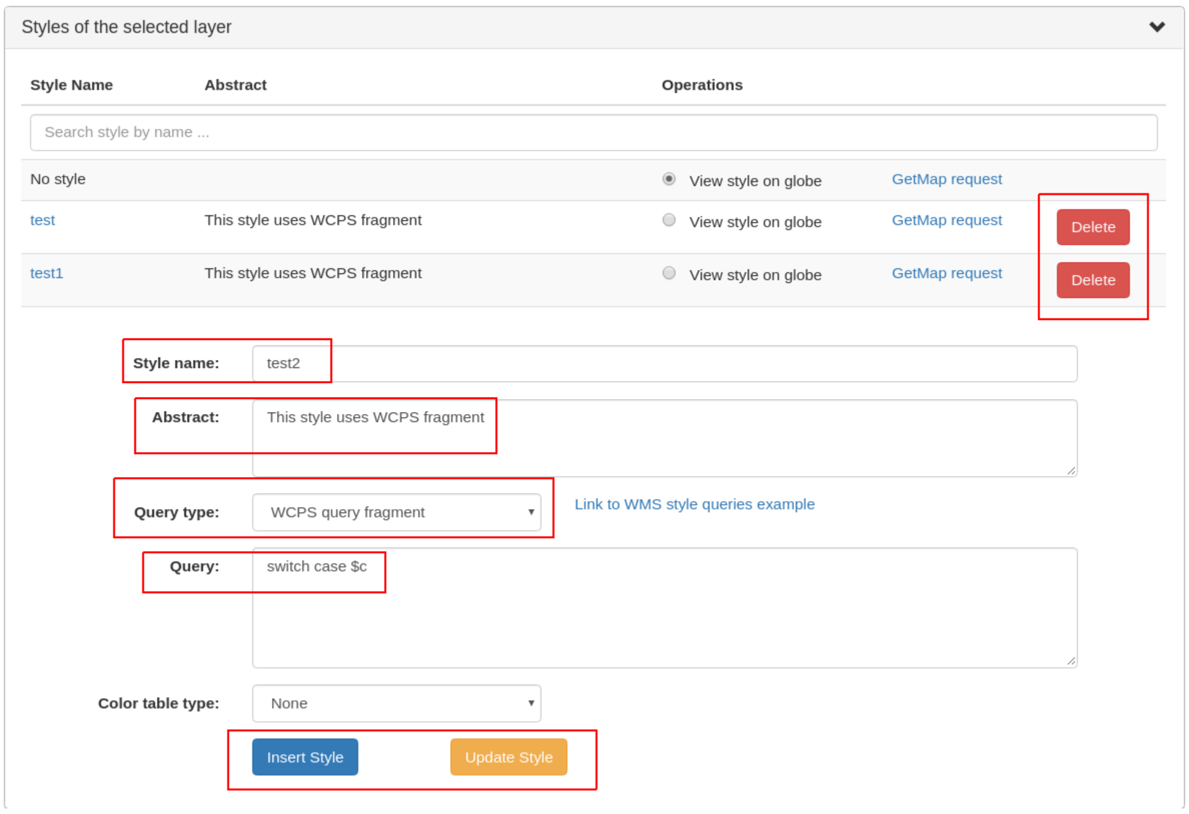


Coverage Id: test_wms_4326

Coverage Extent: lat_min=-44.97500, lon_min=111.97500, lat_max=-8.97500, lon_max=155.97500

Figure 13 Showing/hiding a layer on the map.

Finally, managing WMS styles is possible on this tab. To create a style, it is required to input various parameters along with a rasql or WCPS query fragment, which are applied on every GetMap request if the style is active. Afterwards, click on Insert Style to insert a new style or Update Style to update an existing style of the current selected layer. One can also delete an existing style by clicking on the Delete button corresponding to a style name.


Styles of the selected layer

Style Name	Abstract	Operations		
Search style by name ...				
No style		● View style on globe	GetMap request	
test	This style uses WCPS fragment	○ View style on globe	GetMap request	Delete
test1	This style uses WCPS fragment	○ View style on globe	GetMap request	Delete

Style name: test2

Abstract: This style uses WCPS fragment

Query type: WCPS query fragment

Link to WMS style queries example

Query: switch case $c

Color table type: None

Figure 14 Style management on the DescribeLayer tab.

1.12 Python / Jupyter Notebook

OWSLib is a Python package that helps with programming clients for OGC services such as WCS, WCPS, or WMS. To install it please follow the official installation instructions. Example usage follows below.

```
# Import OWSLib in Python once installed
from owslib.wcs import WebCoverageService

# Create coverage object
my_wcs = WebCoverageService('http://ows.rasdaman.org/rasdaman/ows',
                            version='2.0.1')

# Get list of coverages: ['RadianceColor', 'test_irr_cube_2', ...]
print my_wcs.contents.keys()

# Get Geo-Bounding boxes and native Coverage Reference System (CRS)
print my_wcs.contents['test_irr_cube_2'].boundingboxes

# Get coverage's axis labels
print my_wcs.contents['test_irr_cube_2'].grid.axislabels

# Get coverage's dimension
print my_wcs.contents['test_irr_cube_2'].grid.dimension

# Get Coverage's grid domains (rasdaman domain intervals)
print my_wcs.contents['test_irr_cube_2'].lowlimits
print my_wcs.contents['test_irr_cube_2'].highlimits
```

```
# Get Coverage's offset vectors for geo axes
print my_wcs.contents['test_irr_cube_2'].grid.offsetvectors

# For coverage with time axis get the date time values (year, month, day,
hour, minute, second)
print my_wcs.contents['test_irr_cube_2'].timepositions
```

1.13 NASA WebWorldWind

Simple example to setup a web page with a map from a WMS server using WebWorldWind:

```
<html>
  <head>
    <script
src="https://files.worldwind.arc.nasa.gov/artifactory/web/0.9.0/worldwind.m
in.js"></script>
    <script>
      document.addEventListener("DOMContentLoaded", function(event) {
        WorldWind.Logger.setLoggingLevel(WorldWind.Logger.LEVEL_WARNING);
        var wwd = new WorldWind.WorldWindow("canvasOne");
        var layers = [{
          layer: new WorldWind.BingRoadsLayer(null),
          enabled: true
        }, {
          layer: new WorldWind.CoordinatesDisplayLayer(wwd),
          enabled: true
        }, {
          layer: new WorldWind.ViewControlsLayer(wwd),
          enabled: true
        }];

        for (var l = 0; l < layers.length; l++) {
          wwd.addLayer(layers[l].layer);
        }

        var layerNamesToRequest = ["AvgTemperatureColorScaled"];
        var config = {
          title: "AvgTemperatureColorScaled", version: "1.3.0",
          service: "http://ows.rasdaman.org/rasdaman/ows",
          layerNames: layerNamesToRequest,
          // min Lat, max Lat, min Long, max Long of the requesting layer
          sector: new WorldWind.Sector(-90, 90, -180, 180),
          levelZeroDelta: new WorldWind.Location(36, 36),
          numLevels: 15, format: "image/png", styleNames: "", size: 256
        };

        var wmsLayer = new WorldWind.WmsLayer(config);
        wmsLayer.enabled = true;
        wwd.addLayer(wmsLayer);
      });
    </script>
  </head>
  <body>
      <canvas id="canvasOne" style="width: 100%; height: 100%;"> </canvas>
  </body>
</html>
```

1.14 OpenLayers

Simple example to setup a web page with a map from a WMS server using OpenLayers:

```
<html>
  <head>
     <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/openlayers/3.8.2/ol.css"></lin
k>
     <script
src="https://cdnjs.cloudflare.com/ajax/libs/openlayers/3.8.2/ol.js"></scrip
t>
     <script>
      document.addEventListener("DOMContentLoaded", function(event) {
        var layers = [
          new ol.layer.Tile({
            source: new ol.source.TileWMS({
              url: "https://ahocevar.com/geoserver/wms",
              params: {'LAYERS': 'ne:NE1_HR_LC_SR_W_DR'}
            })
          }),
          new ol.layer.Tile({
            source: new ol.source.TileWMS({
              url: "http://ows.rasdaman.org/rasdaman/ows",
              params: {'LAYERS': 'AvgTemperatureColorScaled'}
            })
          })
        ];
        var map = new ol.Map({
          layers: layers,
          target: 'map',
          view: new ol.View({
            center: [7.5, 53.15178], projection : "EPSG:4326", zoom: 6
          })
        });
      });
     </script>
  </head>
  <body>
    <div id="map" style="width: 100%; height: 95vh"> </div>
  </body>
</html>
```

1.15 Leaflet

Simple example to setup a web page with a map from a WMS server using Leaflet:

```
<html>
  <head>
    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"/>
    <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
    <script>
      document.addEventListener("DOMContentLoaded", function(event) {
        var map = new L.Map('map', {
          center: new L.LatLng(40, 52),
          zoom: 3, attributionControl: true, zoomControl: true, minZoom: 2
        });
        var wmsLayer =
L.tileLayer.wms("http://ows.rasdaman.org/rasdaman/ows", {
          version: '1.3.0', layers: 'AvgTemperatureColorScaled', format:
'image/png'
        });
        map.addLayer(wmsLayer);
      });
    </script>
  </head>
  <body>
    <div id="map" style="width: 100%; height: 100%;"> </div>
  </body>
</html>
```

1.16 QGIS

Instructions can be found in the documentation of QGIS.

1.17 Command-line tools

It's straightforward to make individual OGC WCS / WCPS / WMS requests from the terminal. Examples with curl follow.

- Make a GetCapabilities request:

  ```
  curl "http://ows.rasdaman.org/rasdaman/ows\
  ?service=WCS&version=2.0.1&request=GetCapabilities"
  ```

- Execute a WCPS query with a ProcessCoverages request:

  ```
  curl "http://ows.rasdaman.org/rasdaman/ows" --out test.png --data-
  urlencode \
  'service=WCS&version=2.0.1&request=ProcessCoverages&query=\
  for c in (mean_summer_airtemp) return encode(c, "png")'
  ```

5. Additional Resources

The following resources can be utilized as additional learning material:

- http://doc.rasdaman.org
  http://inspire.rasdaman.org/
  http://tutorial.rasdaman.org/
  http://rasdaman.com
- https://www.opengeospatial.org/standards/wcs
  https://www.opengeospatial.org/standards/wcps
  https://www.opengeospatial.org/standards/wms
- https://processing.code-de.org/rasdaman/
  https://processing.code-de.org/rasdaman/index.html?page=coding-copernicus

PARSEC
ACCELERATOR

Our Partners

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824478.