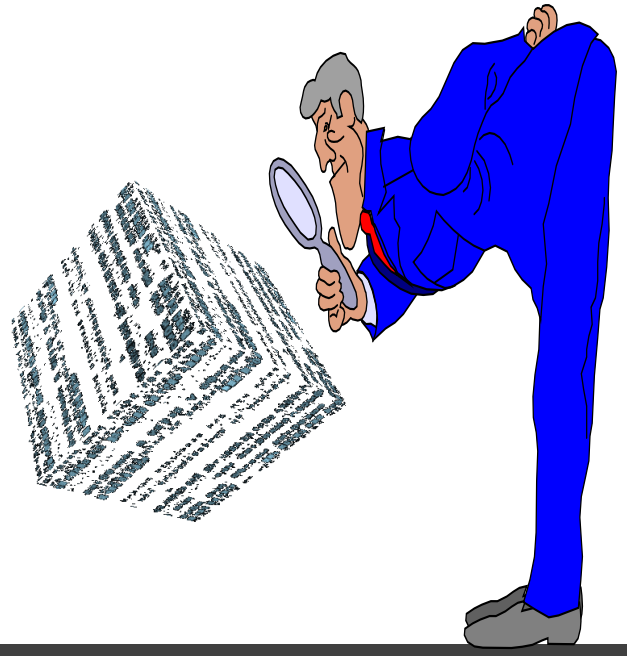
OGC/ISO Coverage Standards

the rasdaman team

Constructor University | rasdaman GmbH

https://constructor.university | rasdaman.com



Why Standards?

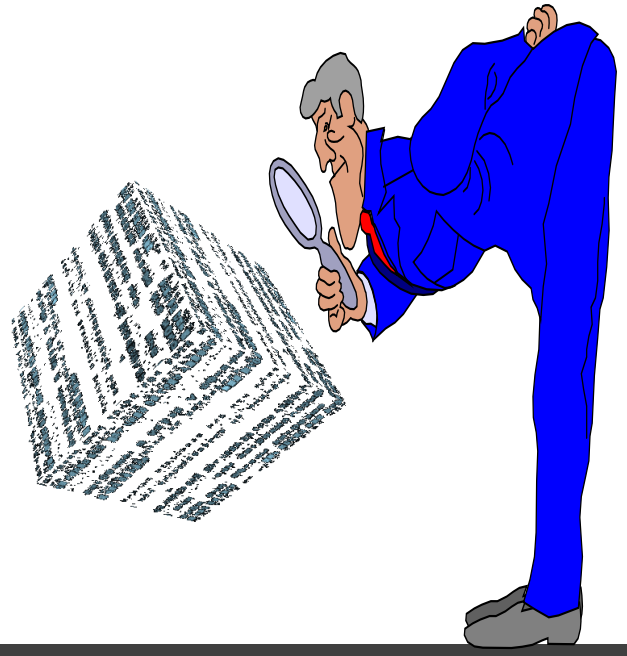
[www.jhchoppers.com]

OGC/ISO Coverage Implementation Schema

the rasdaman team

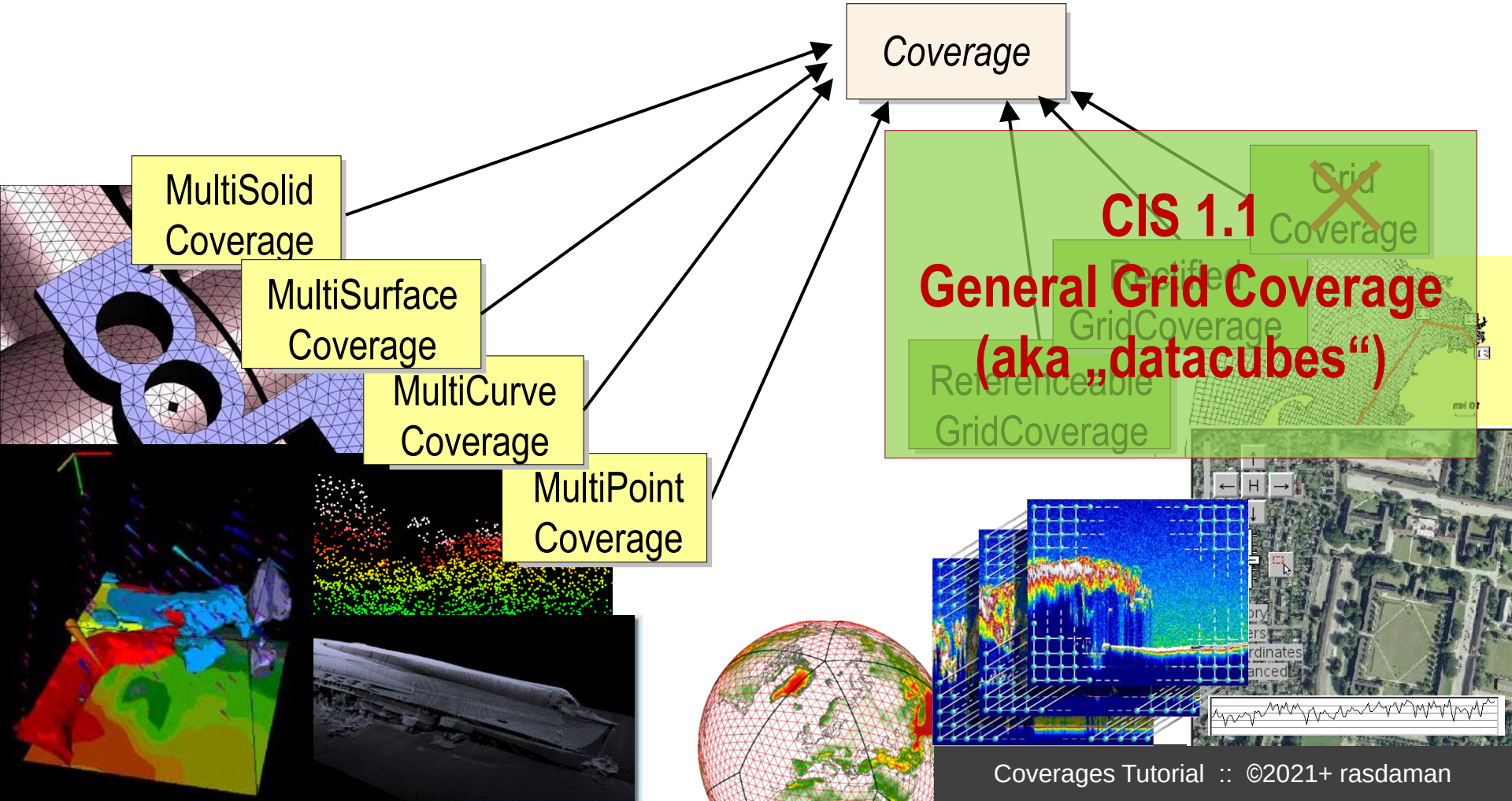
Constructor University | rasdaman GmbH

https://constructor.university | rasdaman.com

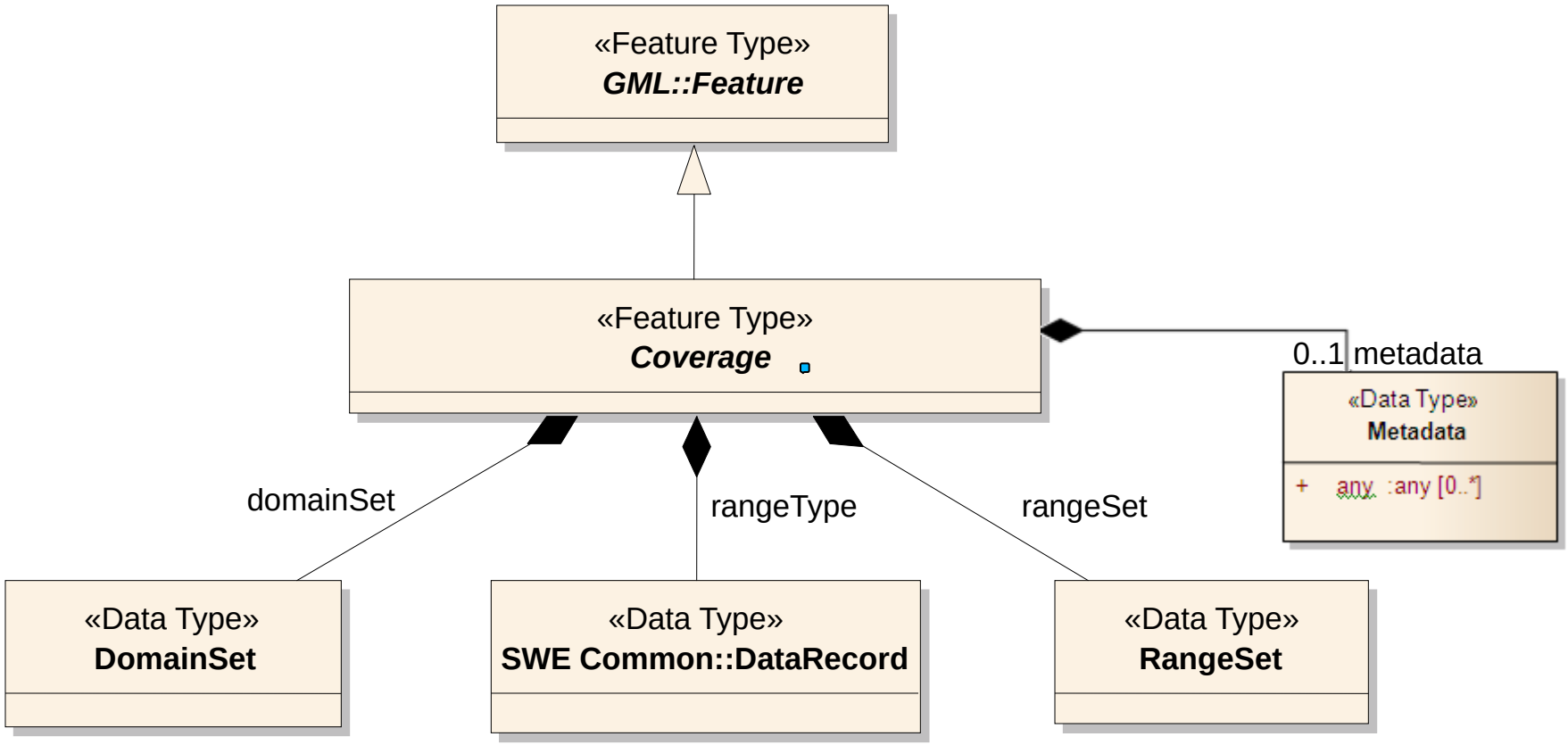


Coverages= Grids + Point Clouds + Meshes

- abstract: OGC Abstract Topic 6, ISO 19123-1
- concrete, interoperable: ISO/OGC Coverage Implementation Schema (CIS, aka GMLCOV)



Coverage Definition



[OGC 09-146r2]

A Simple Coverage, in GML

```
<generalGridCoverage ...  gml:id="CIS_001">

    <domainSet>
        <generalGrid srsName="http://www.opengis.net/def/crs-compound?
                              1=http://www.opengis.net/def/crs/EPSG/0/4979
                             &amp;2=http://www.opengis.net/def/crs/OGC/0/AnsiDate"
            axisLabels="Lat Long h date">
            <regularAxis   axisLabel="Lat"  uomLabel="deg" lowerBound="40"  upperBound="60" resolution="10"/>
            <regularAxis   axisLabel="Long" uomLabel="deg" lowerBound="-10" upperBound="10"  resolution="10"/>
            <irregularAxis axisLabel="h"    uomLabel="m">
                <c>  0</c>
                <c>100</c>
            </irregularAxis>
            <irregularAxis axisLabel="date" uomLabel="d">
                <c>2015-12-01</c>
                <c>2015-12-02</c>
            </irregularAxis>
            <gridLimits srsName="http://www.opengis.net/def/crs/OGC/0/Index4D" axisLabels="i j k l">
                <indexAxis axisLabel="i" lowerBound="0" upperBound="2"/>
                <indexAxis axisLabel="j" lowerBound="0" upperBound="2"/>
                <indexAxis axisLabel="k" lowerBound="0" upperBound="1"/>
                <indexAxis axisLabel="l" lowerBound="0" upperBound="1"/>
            </gridLimits>
        </generalGrid>
    </domainSet>

    <rangeSet>
        <dataBlock>
              <v>01</v> <v>02</v> <v>03</v> <v>04</v> <v>05</v> <v>06</v> <v>07</v> <v>08</v> <v>09</v>
              <v>01</v> <v>02</v> <v>03</v> <v>04</v> <v>05</v> <v>06</v> <v>07</v> <v>08</v> <v>09</v>
              <v>01</v> <v>02</v> <v>03</v> <v>04</v> <v>05</v> <v>06</v> <v>07</v> <v>08</v> <v>09</v>
              <v>01</v> <v>02</v> <v>03</v> <v>04</v> <v>05</v> <v>06</v> <v>07</v> <v>08</v> <v>09</v>
        </dataBlock>
    </rangeSet>

    <rangeType>
        <swe:DataRecord>
            <swe:field name="panchromatic">
                <swe:Quantity definition="http://opengis.net/def/property/OGC/0/Radiance">
                    <swe:uom code="W.m-2.sr-1.nm-1"/>
                </swe:Quantity>
            </swe:field>
        </swe:DataRecord>
    </rangeType>

</generalGridCoverage>
```

A Simple Coverage, in JSON

```
{   "type": "CoverageByDomainAndRangeType",
    "domainSet":{
        "type": "DomainSetType",
        "generalGrid":{
            "type":          "GeneralGridCoverageType",
            "srsName":    "http://www.opengis.net/def/crs/OGC/0/Index2D",
            "axisLabels": ["i", "j"],
            "axis": [{      "type": "IndexAxisType“, "axisLabel": "i", "lowerBound": 0, "upperBound": 2
            },{             "type": "IndexAxisType“, "axisLabel": "j", "lowerBound": 0, "upperBound": 2  }]
        }
    },
    "rangeSet": {     "type": "RangeSetType",
                      "dataBlock": { "type": "VDataBlockType",  "values": [1,2,3,4,5,6,7,8,9] }     },
    "rangeType": {   "type": "DataRecordType",
                      "field":[{     "type": "QuantityType",
                                     "definition": "ogcType:unsignedInt",
                                     "uom": { "type": "UnitReference", "code": "10^0“ }     }]
    }
}
```

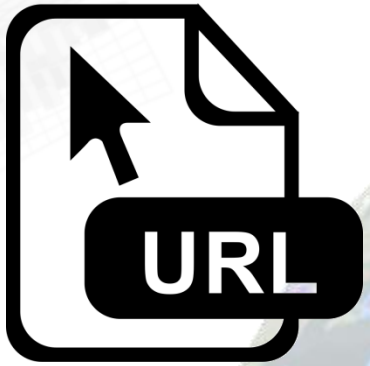
A Simple Coverage, in RDF

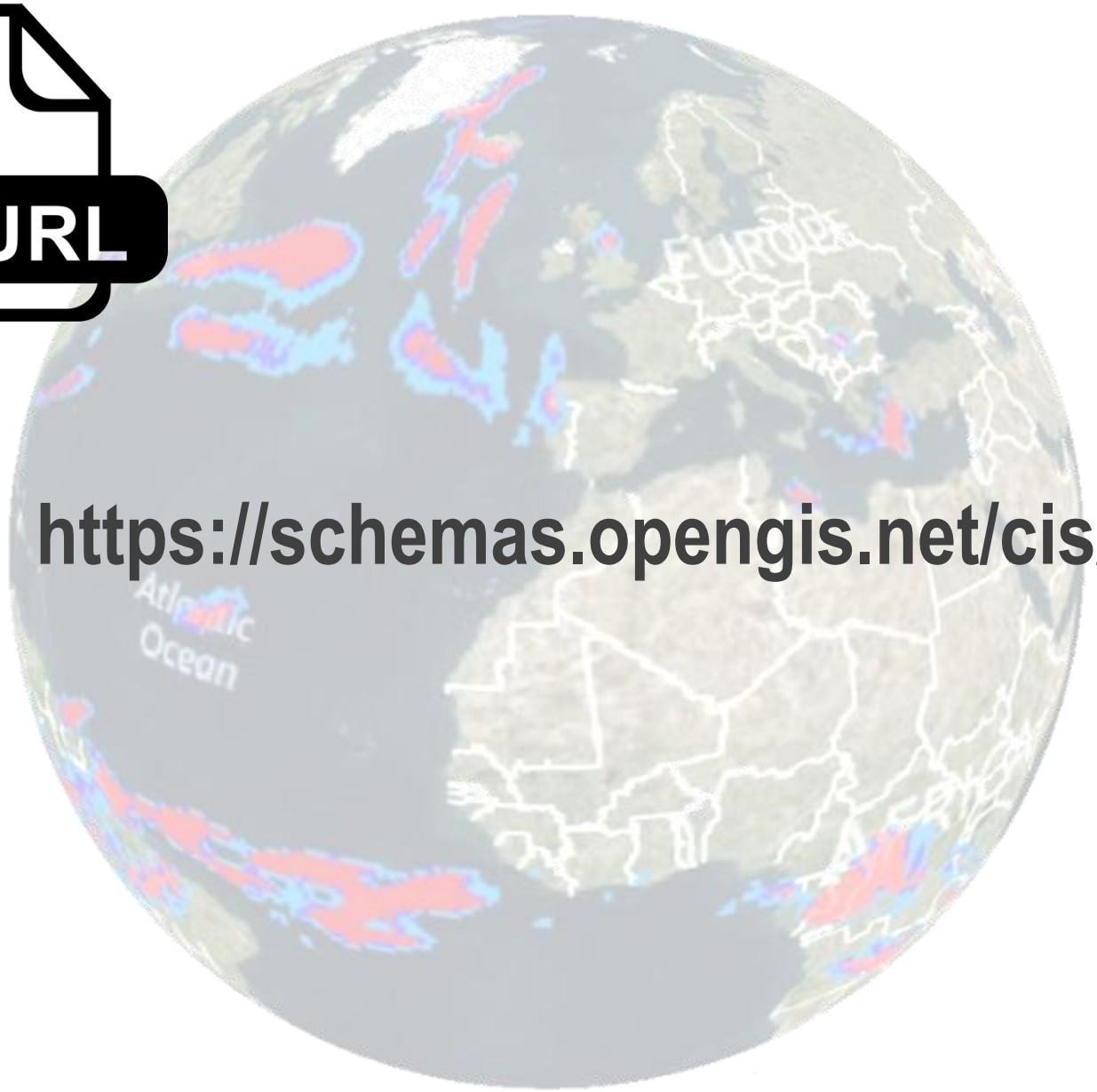
```
<http://www.opengis.net/cis/1.1/examples/CIS_05_2D>
 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.opengis.net/cis/1.1/CoverageByDomainAndRangeType> .

<http://www.opengis.net/cis/1.1/examples/CIS_05_2D>
 <http://www.opengis.net/cis/1.1/domainSet>
 <http://www.opengis.net/cis/1.1/examples/CIS_DS_05_2D> .
<http://www.opengis.net/cis/1.1/examples/CIS_DS_05_2D>
 <http://www.opengis.net/cis/1.1/generalGrid>
 <http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_05_2D> .
<http://www.opengis.net/cis/1.1/examples/CIS_DS_05_2D>
 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.opengis.net/cis/1.1/DomainSetType> .
<http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_05_2D>
 <http://www.opengis.net/cis/1.1/axis>
 <http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_I_05_2D> .
<http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_05_2D>
 <http://www.opengis.net/cis/1.1/axis>
 <http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_J_05_2D> .
<http://www.opengis.net/cis/1.1/examples/CIS_DS_GG_05_2D>
 <http://www.opengis.net/cis/1.1/axisLabels>
 <http://www.opengis.net/cis/1.1/axisLabels0> .
<http://www.opengis.net/cis/1.1/axisLabels0> <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "i" .
<http://www.opengis.net/cis/1.1/axisLabels0> <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.opengis.net/cis/1.1/axisLa
<http://www.opengis.net/cis/1.1/axisLabels1> <http://www.w3.org/1999/02/22-rdf-syntax-ns#first> "j" .
<http://www.opengis.net/cis/1.1/axisLabels1> <http://www.w3.org/1999/02/22-rdf-syntax-ns#rest> <http://www.w3.org/1999/02/22-rdf-sy
```

Coverage Types

- CIS 1.0:
  - [GridCoverage: legacy, faulty, do not use]
  - RectifiedGridCoverage: regular grids
  - ReferenceableGridCoverage: “everything else”
  - Legacy, known to be more involved
- CIS 1.1: addition (no replacement)
  - GeneralGridCoverage: covers more cases, simplified structure
  - Recommended

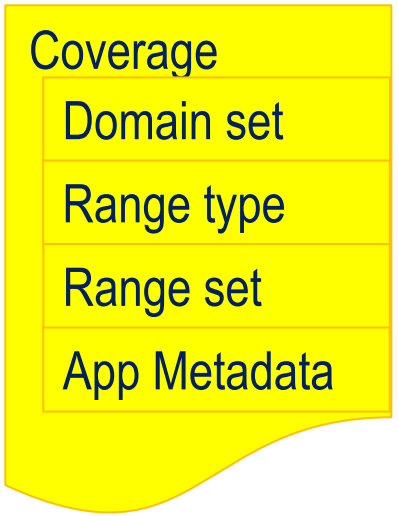




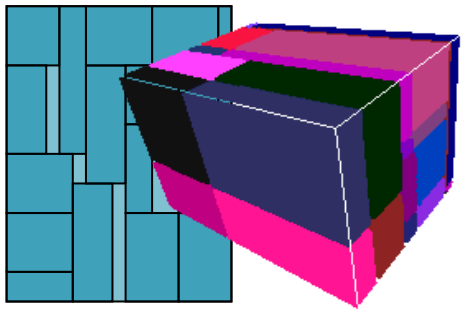
https://schemas.opengis.net/cis/

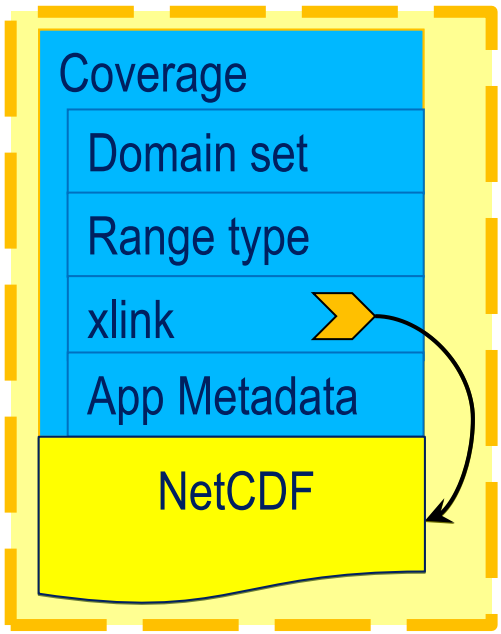
Encoding Coverages

- Single file encoding:
  - Informationally complete: GML, JSON, RDF, …
    - *Caveat: GeoJSON, CovJSON does not work*
  - Further formats: GeoTIFF, NetCDF, JPEG2000, GRIB, …



- Multipart: container(“header” + file1 + file2 + …)
  - Multipart/MIME, zip, GMLJP2, SAFE, GeoPackage, ...
  - Built-in collections / tiling





INSPIRE: Infrastructure for Spatial Information in Europe

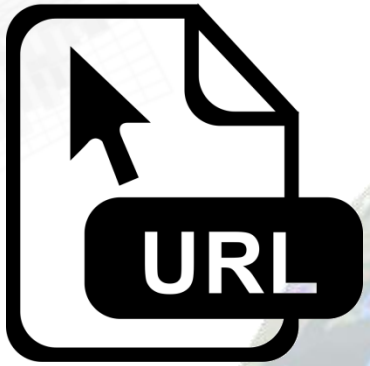
the rasdaman team

Constructor University | rasdaman GmbH

https://constructor.university | rasdaman.com

INSPIRE Coverages

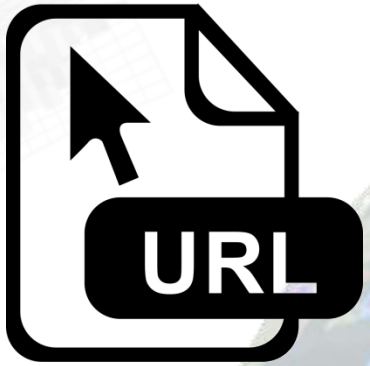
- INSPIRE WCS = OGC WCS-Core
  + OGC WCPS
  + INSPIRE metadata
- INSPIRE metadata modelled CIS-compliant as coverage metadata
- INSPIRE Good Practice: https://inspire-wcs.eu



https://inspire-wcs.eu

→ Demos

→ WCS Client



https://inspire-wcs.eu

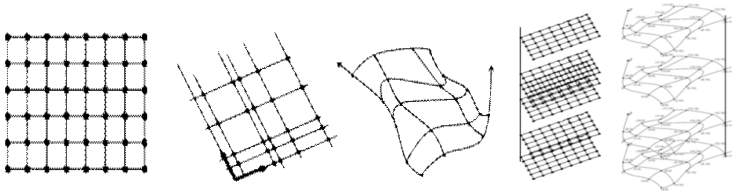
→ Demos

→ WCS Client

Coverage Data Model: Summary

datacubes

- Coverage = regular & irregular grids, point clouds, meshes
- Coverage Implementation Schema(CIS) 1.1
  = backwards-compatible evolution of CIS 1.0



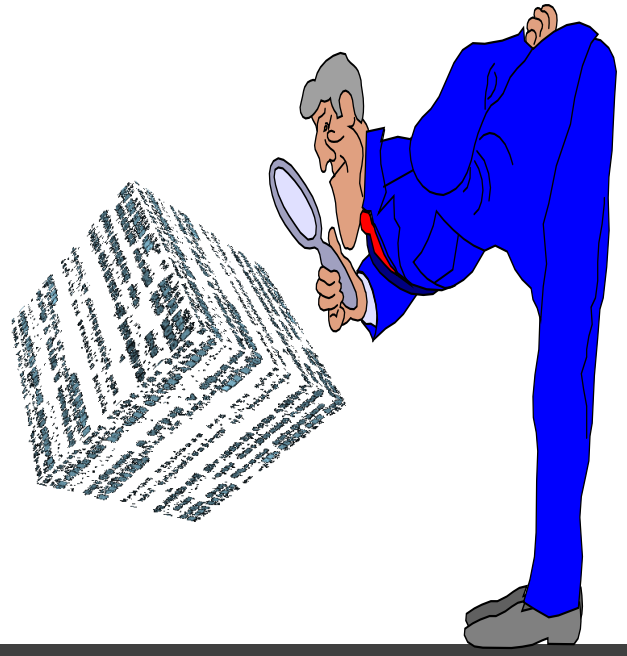
  - See http://schemas.opengis.net/cis/1.1/ for schemas + examples
- *OGC CIS = ISO 19123-2*
  - *Currently, OGC CIS 1.0, under update to include CIS 1.1*

OGC Web Coverage Service (WCS)

the rasdaman team

Constructor University | rasdaman GmbH

https://constructor.university | rasdaman.com

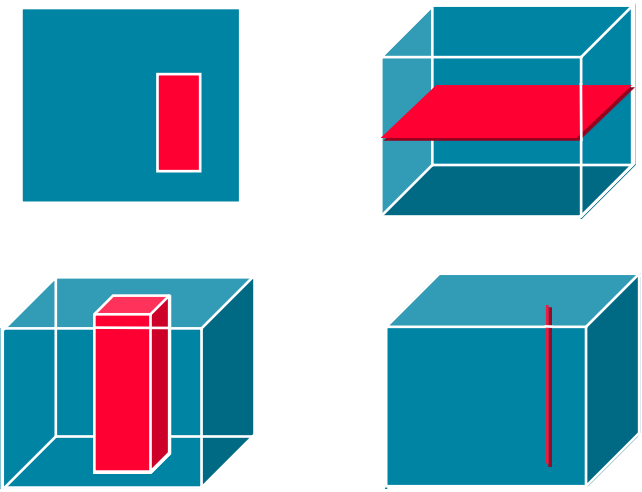


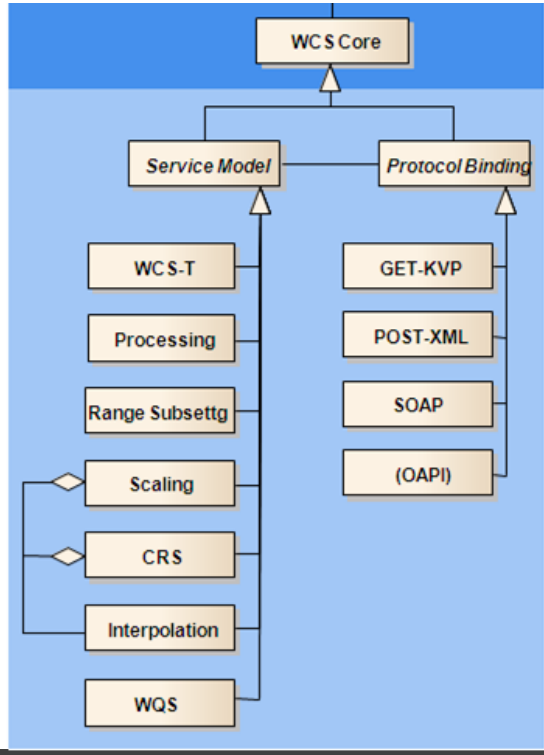
Preamble

- Service model depends on data model
  - but two shoes
- Following WCS, several more service models are emerging
  - OAPI-Coverages, GeoDataCube, EDR, ...
  - Borrowing from WCS, no surprising changes in functionality
- We inspect several of them in turn

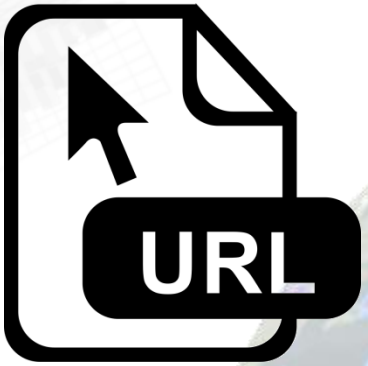
OGC Web Coverage Service (WCS)

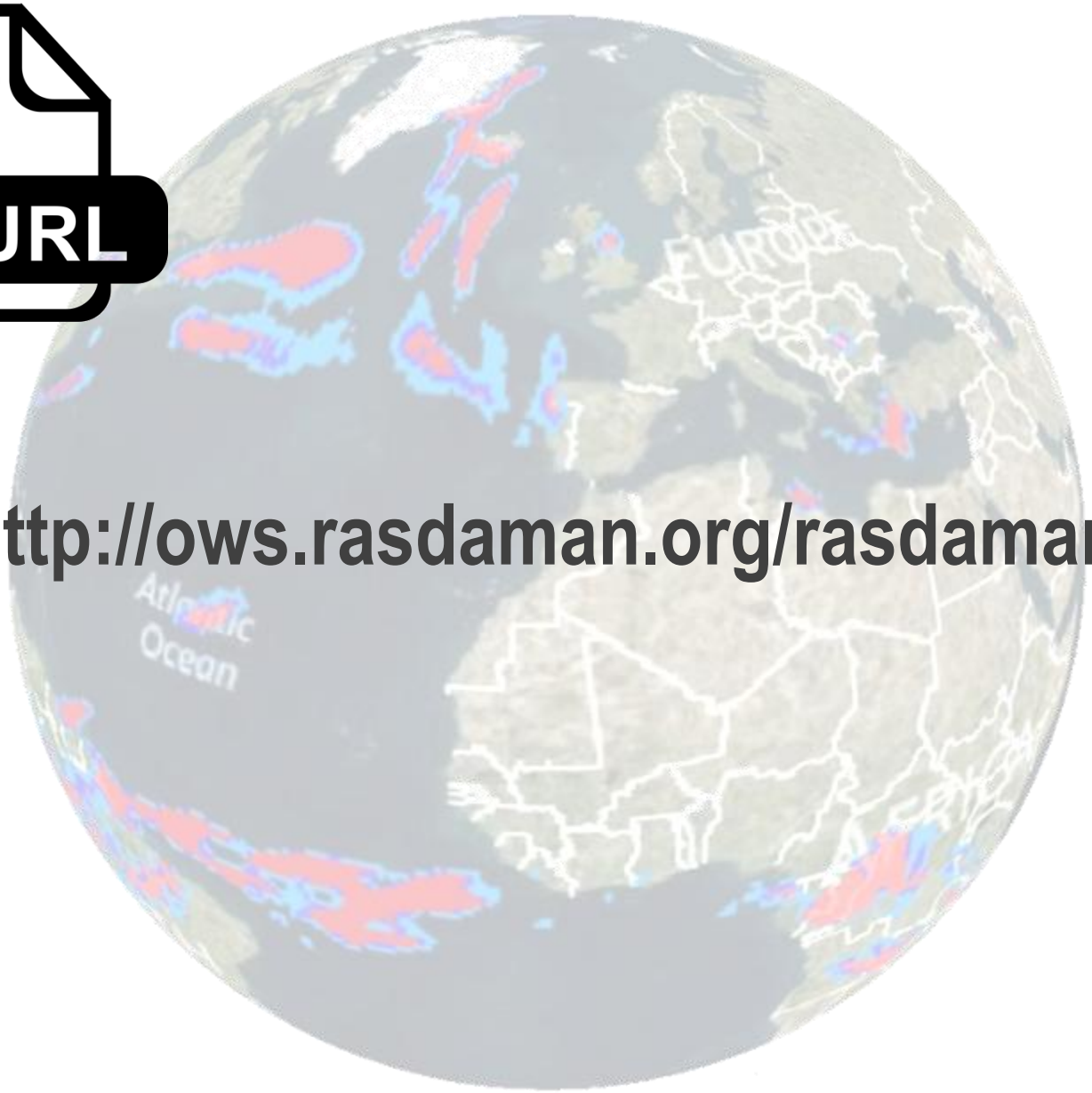
- WCS Core: access to spatio-temporal coverages & subsets
  - Encoding on the fly
  - subset = trim | slice





- WCS Extensions: optional functionality facets
- *rasdaman implements WCS Core & all Extensions*
  - *reference implementation*





http://ows.rasdaman.org/rasdaman/ows

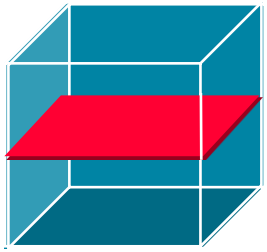
WCS Core *GetCoverage*

- Download a coverage (or a subset thereof), values guaranteed unchanged
- Ex: „*download coverage c001*“

```
http://www.acme.com/wcs ? SERVICE=WCS & VERSION=2.0
    & REQUEST=GetCoverage & COVERAGEID=c001
```

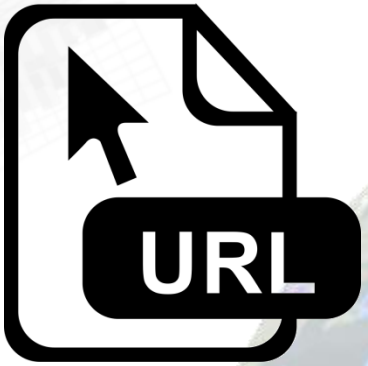
- Ex: „*coverage c001, lat/long cutout, time slice t=2009-11-06T23:20:52*”

```
http://www.acme.com/wcs ? SERVICE=WCS & VERSION=2.0
    & REQUEST=GetCoverage & COVERAGEID=c001
    & SUBSET=Long(100,120) & SUBSET=Lat(50,60)
    & SUBSET=time("2009-11-06T23:20:52")
```



- Ex: “*coverage c001, in GeoTIFF*”

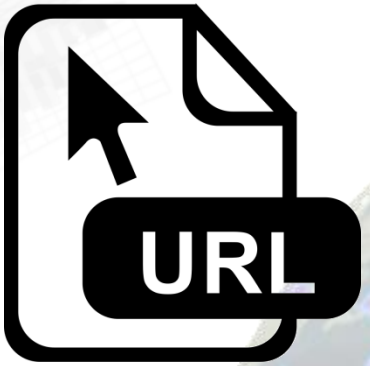
```
http://www.acme.com/wcs ? SERVICE=WCS & VERSION=2.0
    & REQUEST=GetCoverage & COVERAGEID=c001 & FORMAT=“image/tiff“
```



http://ows.rasdaman.org/rasdaman/ows?
SERVICE=WCS&
VERSION=2.0.1&
REQUEST=GetCoverage&
COVERAGEID=NIR&
FORMAT=image/png

OAPI - Coverages

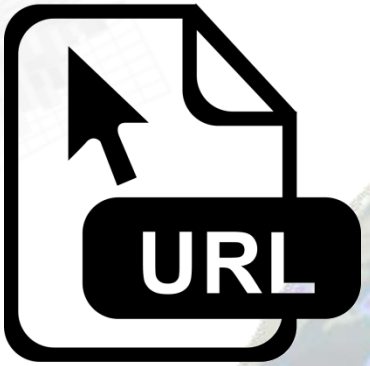
- OGC for unified service interfaces, somewhat RESTful, using YAML
  - Marketing buzzword: API (Application Programming Interface) – still, Web requests
- OAPI-Coverages (OGC workspace, rasdaman-based demo):
  - based on CIS 1.1, like WC(P)S
  - Functionality adopted from WCS
  - ...but new syntax & conventions, some notable detail deviations
    - *„an implementation MAY automatically downsample the coverage to a suitable resolution"*
- Status: Started ~2016, still heavily under development & not adopted
- *Caveat: The specification still has some known shortcomings, is under discussion, and not yet an adopted standard. Hence, the request syntax shown may be subject to unannounced changes*



https://github.com/opengeospatial/ogcapi-coverages

GeoDataCubes

- Recently started SWG for establishing datacube services
- openEO: JSON syntax + visual frontend for different coverage services
  - Started with rasdaman, now ODC,
- OAPI-Processes: „black boxes“ on the server, invoked in requests
- Status: internal discussion documents
- GDC demo video
- *Caveat: The specification still has some known shortcomings, is under discussion, and not yet an adopted standard. Hence, the request syntax shown may be subject to unannounced changes*



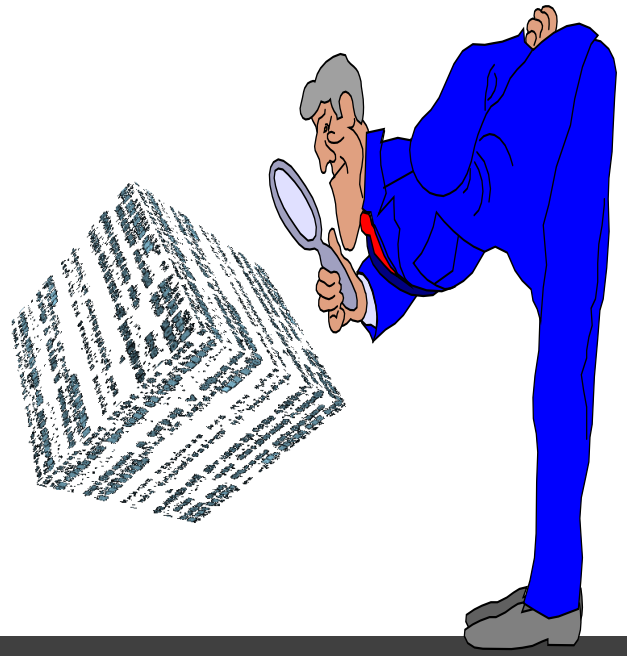
https://testbed19.rasdaman.com

OGC Web Coverage Processing Service (WCPS)

the rasdaman team

Constructor University | rasdaman GmbH

L-sis.org | rasdaman.com

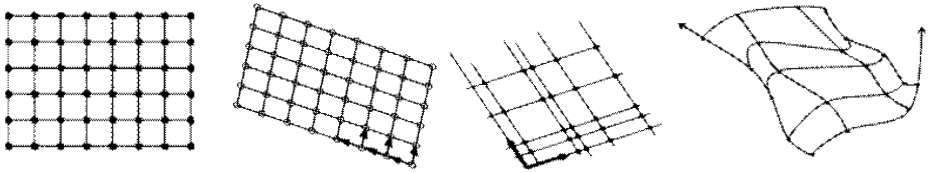


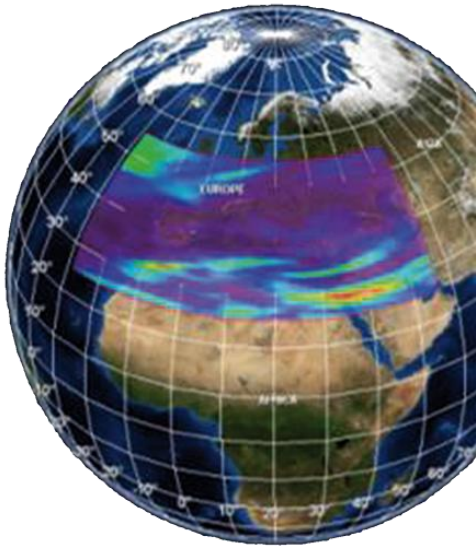
OGC WCPS: Space/Time Datacubes

- Web Coverage Processing Service (WCPS)
  - spatio-temporal datacube analytics language

```
A[ Lat(10.2),Long(8.4),date("2017-12-04")]
```

  - space & time, regular & irregular grids



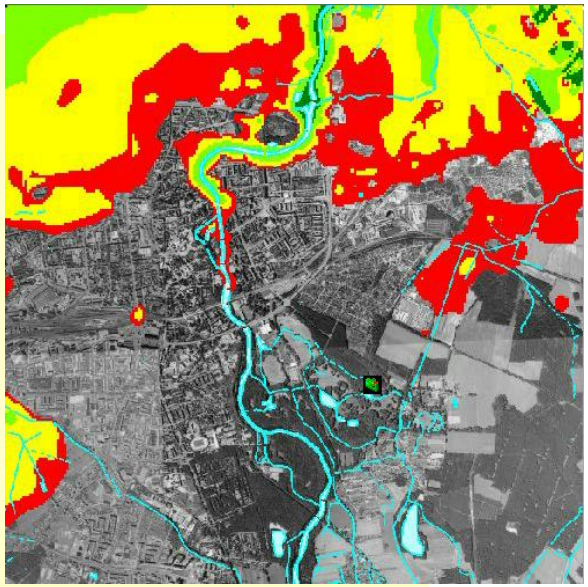


- "From MODIS scenes M1, M2, M3: difference red & nir, as TIFF"
  - "…but only those where nir exceeds 127 somewhere"

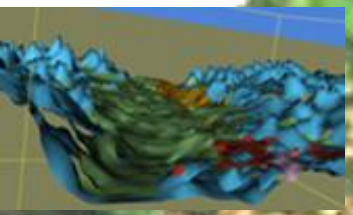
```
for $c in ( M1, M2, M3 )
where some( $c.nir > 127 )
return encode( $c.red - $c.nir, "image/tiff" )
```

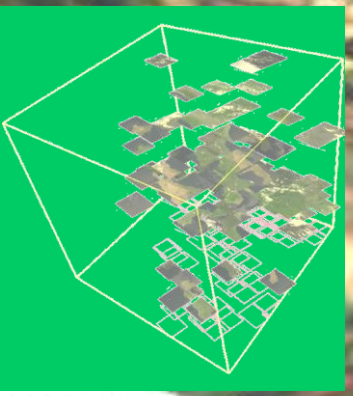
WCPS Emulating WMS

```
for $p in (OrthoPhoto),
    $wl in (WaterLines), $wa in (WaterAreas),
    $d in (DEM)
return
    encode(  (unsigned char) (
          $p * { 1, 1, 1 }
        overlay
          $wl * { 0, 128, 255 }
        overlay
          $wa * { 191, 255, 255 }
        overlay
          switch $d
          case $d > 260 return { red:255, green:0, blue:0 }
          case $d > 262 return { red:0, green:255, blue:0 }
          case $d > 264 return { red:0, green:0,  blue:255 }
          default       return { red:0, green:0, blue:0 }
          end
       ),
      "image/png" )
```



WCPS: Elevation & Image Fusion





```
for $s in (SatImage), $d in (DEM)
where $s/metadata/@region = "Glasgow"
return
  encode(
    struct {
      red:   (char) $s.b7[x0:x1,x0:x1],
      green: (char) $s.b5[x0:x1,x0:x1],
      blue:  (char) $s.b0[x0:x1,x0:x1],
      alpha: (char) scale( $d, 20 )
    },
    "image/png"
  )
```

[JacobsU, Fraunhofer; data courtesy BGS, ESA]

WCPS via Command Line

- Request type = Processing, so URL schema is:

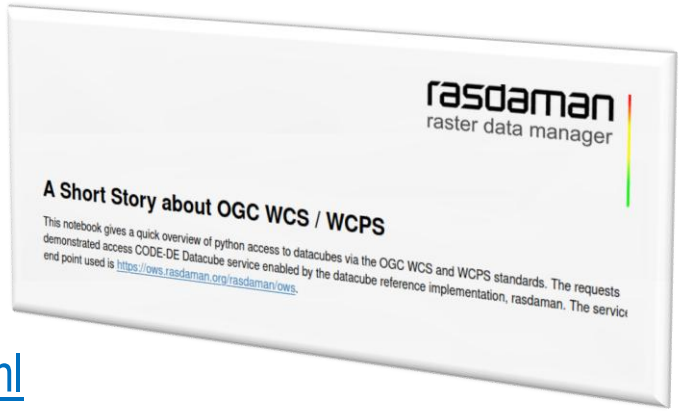
```
https://standards.rasdaman.com/rasdaman/ows?
        SERVICE=WCS&
        VERSION=2.0.1&
        REQUEST=Processing&
        query={your-query-here-escaped}
```

- Any URL resolution tool, such as **`curl`**

- Ex:

```
$ 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")'
```

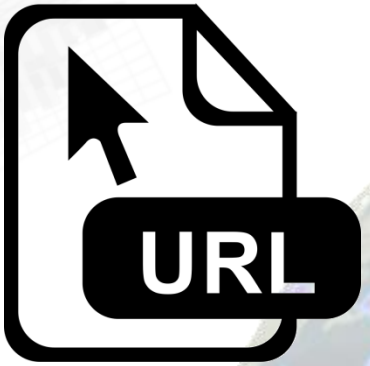
WCPS in Python & R



- Use http requests
  - https://standards.rasdaman.com/demo_jupyter.html
  - https://nbviewer.org/github/earthserver-eu/INSPIRE-notebooks/blob/master/index.ipynb

- Ex:

```
from IPython.display import Image
import requests
query = '''
for $c in (S2_L2A_32631_B12_20m),
    $d in (S2_L2A_32631_B08_10m),
    $e in (S2_L2A_32631_B03_10m)
let $cutOut := [ ansi( "2021-04-09" ), E( 670000:730000 ), N( 4990220:5015220 ) ]
return
  encode(
      scale( { red: $c[ $cutOut ]; green: $d[ $cutOut ]; blue: $e[ $cutOut ]}, { E:"CRS:1"(0:599) } ) / 15.0,
      "image/png")
'''
response = requests.post( service_endpoint, data = {'query': query}, verify=False )
Image( data=response.content )
```



standards.rasdaman.com/demo_wcps.html

inspire.rasdaman.com

WCPS Status

- WCPS 1.0 (what we discuss)
  - All of Tomlin's Map Algebra, plus image / signal processing up to ~DFT
  - Problem: established before CIS
- WCPS 1.1: aligned with CIS 1.1, more functionality; backwards compatible
  - Metadata queries embedded
- Adopted by ISO as 19123-3
  - Adopted by EU INSPIRE
  - under adoption vote by OGC as Abstract Topic 6 Part 3

WCPS 1.1 Impressions

- 2D WGS84 grid with several allowed interpolation methods

```
domain set
  crs "EPSG:4326" with
  Lat  regular (10:30) resolution 0.01,
  Long regular (10:30) resolution 0.01,
  interpolation nearest, linear, quadratic, cubic
```

- Irregular and index axes

```
domain set
  crs "EPSG:4326+OGC:unixTtime" with
  Lat  regular (10:30) resolution 0.5,
  Long regular (10:30) resolution 0.5,
  date irregular ( "2017-01-01", "2017-02-01",  "2017-03-01", "2017-04-01",
    "2017-05-01", "2017-06-01", "2017-07-01", "2017-08-01",  "2017-09-01" ),
  band index (1:32)
```

WCPS 1.1 Impressions

- Build new coverage

```
coverage GeneralGridCoverage Half
range set (unsigned char) $C / 2
```

...or:

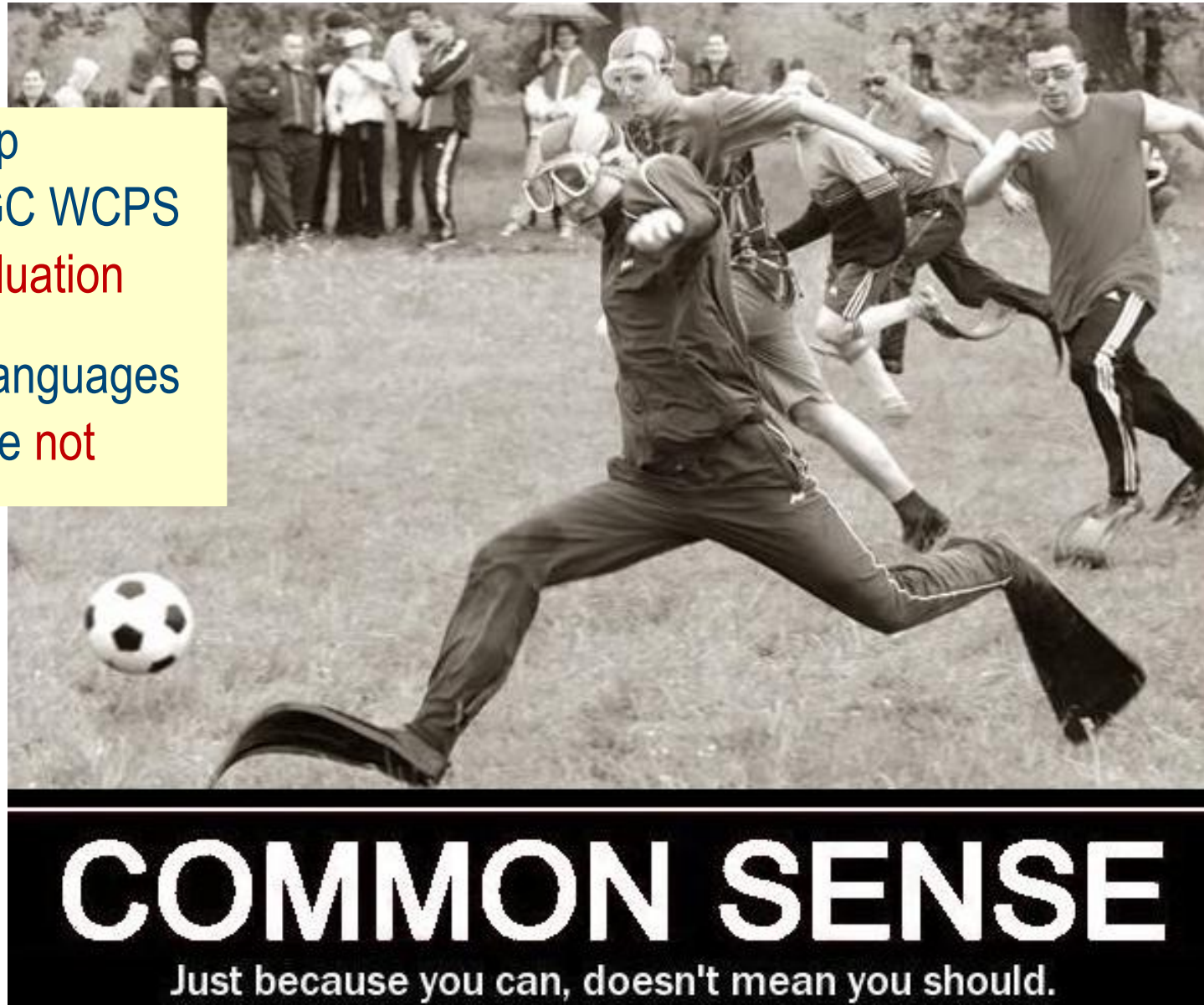
```
(unsigned char) $C / 2
```

- Build histogram of n-D coverage

```
coverage GeneralGridCoverage histogram
domain set crs OGC:Index1D with bucket index (0:255)
range type b : unsigned long
range set count( $C.blue = bucket )
```

„Ship Code to Data“ -- *What Code to Ship?*

- High-level, crisp ISO SQL & OGC WCPS are safe in evaluation
- programming languages (like python) are not



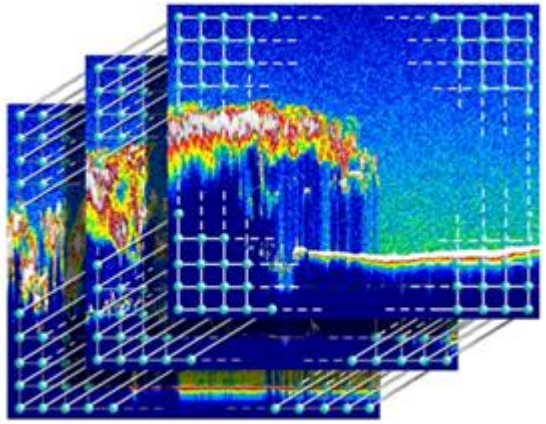
[pinterest]

„But I have to Learn a Language!“

- Tutorials: https://earthserver.world/wcs
- Chatbot support: https://ai-cu.be/chatbot

Summary

- WCPS = ISO + OGC + INSPIRE geo datacube analytics language
- High-level = zero-coding
- Server can optimize, parallelize, ...
- „Swiss army knife“ for analytics
  - Since recently: AI (see later)



- More info @ OGC: https://myogc.org/go/coveragesDWG

For later: Sample Tasks

- Submission URL:
  - https://ows.rasdaman.org/rasdaman/ows?SERVICE=WCS&VERSION=2.0.1&REQUEST=Processing&query={your-query-here-escaped}
- Extract timeslice from AvgLandTemp datacube:
  - for $c in (AvgLandTemp)
    return encode($c[ansi("2014-01")], “png”)
- Extract timeline from AvgLandTemp datacube:
  - for $c in (AvgLandTemp)
    return encode($c[Lat(53.08), Long(8.80)], “csv”)
- Minimum temperature in January 2014:
  - for $c in (AvgLandTemp)
    return min($c[ansi("2014-01")])

For later: Sample Tasks

- Submission URL:
  - https://standards.rasdaman.com/rasdaman/ows?SERVICE=WCS&VERSION=2.0.1&REQUEST=Processing&query={your-query-here-escaped}
- Mean summer temperature inside polygon:
  - for $c in (mean_summer_airtemp)
    return
      encode(
        clip($c,
          POLYGON((-12.3829 132.0117, -33.4314 120.4102, -18.8127 148.5352,
                     -22.7559 118.4766, -36.3151 143.7891)
)
        ),
        "image/png",
        "{\"nodata\": [0] }"
)

For later: Sample Tasks

- Submission URL:
  - https://standards.rasdaman.com/rasdaman/ows?SERVICE=WCS&VERSION=2.0.1&REQUEST=Processing&query={your-query-here-escaped}
- In the various demos, grab the WCPS queries, run them in the WCPS console
  - https://standards.rasdaman.com/demo_wcps.html