Data discovery and basic access

Massimo Argenti
massimo.argenti@esa.int
Enable multidisciplinary scientists to **access** and **study** data from **multiple domains** for “system level” research by providing solutions and **guidelines for the RIs common needs**

Multiple data producers
Multiple data consumers
Discovery and Basic Access with ENVRI

- Discover heterogeneous data at different places and in different catalogues
  - Distributed measurements and monitoring
    - physical, chemical and biological parameters
  - Laboratories and experimental facilities
    - in fixed monitoring stations
    - on research vehicles, ships, floats and buoys
    - from aircraft and satellites
  - A variety of data
    - heterogeneous in format
    - primary and processed data
  - Analytical and modeling platforms
    - data exchange and integration
    - high performance computing and Grid services
    - e-Laboratories
Behind the data discovery-access

- A federation of catalogues:
  - One at the level of the portal/client containing the metadata at series collection level
  - Many catalogues at the level of the federated resources containing the metadata at dataset product level

- All the catalogues can be accessed using OGC OpenSearch protocol, a collection of technologies allowing websites and search engines to publish search results in a standard and accessible format (http://www.opensearch.org/)

- Data/products remain at their original location, i.e., where the Data Provider stores them; the Catalogues provide the user with the link to directly access the data
Outline

- **Syntactic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Semantic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Notes for the developer**
  - OpenSearch and the RDF model
  - Machine-to-machine syntactic demo
  - The semantic framework in depth
Outline

- Discovery and Access with ENVRI: an introduction

  - Syntactic Discovery
    - A discovery and basic access example
    - What’s behind: high level architecture and basic flow
    - Demo

  - Semantic Discovery
    - A discovery and basic access example
    - What’s behind: high level architecture and basic flow
    - Demo

- Notes for the developer
  - OpenSearch and the RDF model
  - Machine-to-machine syntactic demo
  - The semantic framework in depth

17/06/14 PIRE Workshop
Interferograms computed from data (either on demand computation or discovery of previously generated products)

Query of heterogeneous data based on geo-spatial and temporal criteria defined by the user

Satellite data

In-Situ data
ENVRI data are organised in collections or series each of which containing datasets (products)

- Series discovery: the user submits a query to discover the available series matching the user search criteria
- Dataset discovery: for each series, the datasets available at the Geospatial repositories are discovered
- Dataset access: for each discovered dataset, a download link is provided (if allowed by the provider)
Syntactic Discovery – a discovery and basic access example

Select the catalogue

Insert the search text string

Insert Start Date and Stop Date

Set the bounding box as desired

Collections of dataset corresponding to the search criteria are listed here

Datasets belonging to the selected collection (access URL is also provided)
Syntactic Discovery - What’s behind: high level architecture

- ENVRI uses a federation of distributed catalogues inherited from GENESI-DEC
- 2-steps discovery:
  - One or more Aggregator Nodes contain the metadata at series collection level
  - Each federated resources exposes a catalogue containing the metadata at dataset product level
- All the catalogues can be accessed using OGC OpenSearch protocol, a collection of technologies allowing websites and search engines to publish search results in a standard and accessible format
- User can directly query the Catalogues using OpenSearch or through Clients, as the ENVRI webportal
- Data/products remain at their original location, i.e., where the Data Provider stores them; the Catalogues provide the user with the link to directly access the data
Syntactic Discovery - What’s behind: the RDF metadata model

- The metadata model is based on RDF (Resource Description Framework)
- The RDF model:
  - A “Series” section: includes information shared by all the datasets belonging to that series.
  - One or more “Dataset” sections: includes information shared by all the data of the dataset (a dataset represents an identifiable collection of data)
  - Can be expanded as needed according to the specific needs of the communities
Data Providers should:

- Provide online access to their data/products (restricted access if needed)
- Create a metadata catalogue for these data/products – at least metadata useful/needed for discovery – domain specific metadata can be added
- Set up OpenSearch Interfaces for the catalogues
- Register the catalogues endpoints to the ENVRI aggregator nodes
Outline

- **Syntactic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Semantic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Notes for the developer**
  - OpenSearch and the RDF model
  - The semantic framework
  - Demo
Semantic Discovery – A discovery and basic access example

Browse Ontology Terms

Collections of dataset corresponding to the search criteria are listed here

Datasets belonging to the selected collection (access URL is also provided)

Geo reference zone of the selected dataset
Semantic Discovery – what’s behind: Ontology Example
An Ontology is a formal and explicit description of concepts (or classes) in a specific domain. An ontology is made of:

- Concepts (classes)
- Relations between concepts (e.g. ‘is-a’ or ‘instanceOf’)
- Concepts attributes (slots or roles or properties) describing properties of classes or instances
- Restrictions on attributes (facets or role restrictions)

A Knowledge Base is an ontology together with a set of individual instances of classes.

The big task in ENVRI will be the creation of an Ontology that should merge the concepts and relations of different domains like DRs and new Infrastructure.
Semantic Discovery – what’s behind: Semantic Web

Tim Berners-Lee originally expressed the vision of the semantic web as follows:

“I have a dream for the Web [in which computers] become capable of analysing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.”

- The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

- The Semantic Web is a vision of data that is understandable by machines, so computers can perform more of the tedious work involved in finding, combining, and acting upon data on the web.
The Semantic Web Stack is a hierarchy of languages and technologies used to create the Semantic Web:

- the lower layers comprise well-known technologies from the classical hypertext web (e.g. Unicode and XML)
- the middle layers comprise technologies for enabling semantic web applications to be built (e.g. RDF and OWL)
- the top layers contain those technologies required to bring the semantic web to full fruition.
Semantic Discovery – what’s behind: RDF & OWL

- RDF and OWL are relatively simple things compared to AI and they offer:
  - a simple way to express and store metadata
  - a way to “structure” and characterize the terms
  - means to make some inference within a restricted framework

and that is it!

- The atomic element in an RDF description is the triple

- An \((s,p,o)\) triple can be viewed as a labeled edge in a graph (Subject, Predicate, Object)
  - i.e., a set of RDF statements is a directed, labeled graph
    - both “objects” and “subjects” are the graph nodes
    - “properties” are the edges
Inference is the process of deriving new knowledge starting from the relevant (domain) ontology and the available knowledge (available from the knowledge base). The process is performed by a Reasoner.

The inference should be performed when new semantic information’s are inserted in the tag repository or on the fly as first step in a discovery operation.

Global inference process should be centralised taking into account the new tagged information.
Semantic Discovery – what’s behind: Inference

From the ontology we can infer → SNOW is a Land, → ICE is a Land
From the statement ‘The datasetX is-a SNOW’, looking at the ontology we can infer → ‘The datasetX’ is-a SNOW & ICE → ‘The datasetX’ is-a LAND
Set up a Semantic solution to close the resource discovery gap.

Adding Semantics considering:
- Different domain ontologies (i.e. with the new External Infrastructure)
- Multi-domain and multilingual context
- Semantic links between data resources (possibly of different owners - from different domains)

The ENVRI User Community must play a proactive role with data owner tagging their own data or data user consumer tagging after the data use.

Users are normally tagging resources belonging to their domain of interest, so they are domain expert users.
The following thesaurus have been introduced to describe, annotate and discover the ENVRI resources:

- GEMET (GEneral Multilingual Environmental Thesaurus) provides a user friendly parameter discovery interface for the European Environment Information and Observation Network (EIONET). It makes use of SKOS, (Simple Knowledge Organisation System) and also the metadata registries standard, ISO 11179.

- SBA (Social Benefit Areas) GEO Group and observation is constructing GEOSS on the basis of a 10-year implementation plan for the period 2005 to 2015.

- GCMD (Global Change Master Directory) science keywords list is a comprehensive directory of information about Earth science data, including the oceans, atmosphere, hydrosphere, solid earth, biosphere and human dimensions of global change. (NASA)
Semantic Discovery – what’s behind: ENVRI tagging

- In addition to data discovering, a user in ENVRI can:
  - Provide a data resource URI and associate semantic metadata to the resource
  - Semantic metadata are supposed to be represented by concepts from an ontology
  - Users are normally tagging resources belonging to their domain of interest, so they are domain expert users
Semantic Discovery – what’s behind: high level architecture

- **Tagging Client** and **Discovery Client** are the portal UI where the user interacts with the semantic framework.

- **Discovery Processor** is the component devoted to discovery resources using a predefined set of ontology terms.

- **Tagging Processor** is the component devoted to tag a resource discovered linking the tag to the user logged.

- **Harvester** is the component devoted to populate the knowledge base extracting the information from catalogues repositories.

- **Semantic Processor** is the component that manage the interface with the ontology DB and all the semantic operations.
Syntactic Demo

- A syntactic query from the portal

Access [http://portal.envri.eu](http://portal.envri.eu)

- Insert “SAR Etna” in the free text field, select Sicily area as the bounding box and press “Search”
- Select one of the tiff files from the result list and download it locally

- Insert “GPS” in the free text field, select Sicily area as the bounding box and press “Search”
- Select one of the text files from the result list and download it locally
Semantic Demo

A semantic query from the portal

- Select the “Semantic” option and browse the categories to identify data of interest.
- Show discovery, access and positioning of data for generic terms
- Browse specific semantic term like $TEST\_CASE \rightarrow ISLAND\_VOLCANO$
- Select one of the files from the result list and download it locally
- Browse specific semantic term like $GCMD \rightarrow SOLID\_EARTH \rightarrow VOLCANOES \rightarrow VOLCANIC\_ASH/\_DUST$
- Select one of the files from the result list and download it locally
Outline

- **Syntactic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Semantic Discovery**
  - A discovery and basic access example
  - What’s behind: high level architecture and basic flow
  - Demo

- **Notes for the developer**
  - OpenSearch and the RDF model
  - Machine-to-machine syntactic demo
  - The semantic framework in depth
The catalogues of the different repositories expose an OpenSearch-based interface by which data can be discovered and accessed through external applications.

OpenSearch is a collection of technologies allowing websites and search engines to publish search results in a standard and accessible format.

Search engines are described through OpenSearch Description Documents.
Notes for the developer – OpenSearch Description Document

The URL element:
- provides in the template attribute the URL and format to be used by clients to query the search engine. OpenSearch-defined parameters are used as placeholders (i.e. searchTerms, startPage)
- can occur more than once, since there is one for each format in which the results are returned: the type attribute is the MIME type of such format.

In ENVRI, the Aggregator Nodes as well as each geospatial repository site search engines are described as OpenSearch Description Documents
The metadata model is based on RDF - Resource Description Framework

The RDF model:

- **A “Series” section:** includes information shared by all the datasets belonging to that series.

- **One or more “Dataset” sections:** includes information shared by all the data of the dataset (a dataset represents an identifiable collection of data)

- **Can be expanded as needed according to the specific needs of the communities**
Example RDF: series section

```xml
<dc lite4g:Series rdf:about="http://dr-site.esrin.esa.int/catalogue/genesis/CultureMERIS/rdf">
  <dc:identifier>Culture MERIS</dc:identifier>
  <dc:title>Culture MERIS</dc:title>
  <dc:abstract>
    Culture-MERIS products are targeting particular land activities in Europe and Africa, for which a quick and updated overview of the landscapes is needed. To access the processing chain that has been established during the ESA’s DUE GlobCover project by MEDIAS-France (see RD-1, RD-2). Culture-MERIS project uses the on-line archive (Rolling Archive) of ESRIN (PDHS-E) and Kiruna (PDHS-E) stations. Every week (on Wednesdays) an updated MERIS FR (300 m.) bottom-up cloud-free composite is provided, based on data that have been acquired during the previous week (Monday to Sunday acquisitions). Therefore Culture-MERIS provides weekly (Monday to Sunday acquisitions) cloud-free spectral reflectance composites as derived from MERIS FR (300 m.) data acquired over Europe and Africa and processed using the GlobCover processing chain.
  </dc:abstract>
  <eop:platform>Envisat</eop:platform>
  <ical:dtstart>2011-03-06T23:00:00.000Z</ical:dtstart>
  <ical:dtend>2011-04-09T22:00:00.000Z</ical:dtend>
  <dct:spatial>POLYGON((-25,-35,-25 75,65 75,65 -35,-25 -35))</dct:spatial>
  <dc:subject>Land</dc:subject>
  <dc:rights>Free Usage with ESA Credits</dc:rights>
  <dc:format>HDF</dc:format>
  <dclite4g:resolution>300 meters</dclite4g:resolution>
  <eop:processingLevel>Level 3</eop:processingLevel>
  <dct:extent>1135</dct:extent>
  <dct:modified>2012-02-17T15:10:30.786Z</dct:modified>
  <eop:sensorType>OPTICAL</eop:sensorType>
  <dc:publisher rdf:resource="http://www.esa.int"/>
</dclite4g:Series>
```
Querying ENVRI-like systems programatically

- First level query towards the Aggregator Node(s)
- Second level query towards the geospatial repositories matching the search criteria in the first level query
- Both queries are enabled by a Catalogue Access Service based on OpenSearch
Notes for the developer – Querying ENVRI-like systems programatically

1. Aggregator Node description request
   http://catalogue.genesi-dec.eu/search/description

2. Aggregator Node OpenSearch Description Document
   in the template attributes shows how to query the Aggregator Node, so you can:

   e.g. http://catalogue.genesi-dec.eu/search/rdf/?q=ETNA

3. Submit query to the Aggregator Node
Notes for the developer – Querying ENVRI-like systems programatically

1. Aggregator Node description request
   http://catalogue.genesi-dec.eu/search/description

2. Aggregator Node OpenSearch Description Document
   in the template attributes shows how to query the Aggregator Node, so you can:
   e.g. http://catalogue.genesi-dec.eu/search/rdf/?q=ETNA

3. Submit query to Aggregator Node

4. Series metadata file
   in the atom:href attribute provides the URL of the OpenSearch Description Document of the series at issue

17/06/14 PIRE Workshop
1. Opensearch description request for the series of interest
   e.g. http://dr-ext.genesi-dec.eu/catalogue/genesi/Etna_SAR/description/

2. OpenSearch Description Document for the series of interest
   in the template attributes shows how to query datasets from the series of interest

3. Submit (refined) dataset query
   e.g. http://dr-ext.genesi-dec.eu/catalogue/genesi/Etna_SAR/rdf/?start=2007-06
Notes for the developer – Querying ENVRI-like systems programatically

1. OpenSearch description request for the series of interest
2. OpenSearch Description Document for the series of interest
   e.g. http://dr-ext.genesi-dec.eu/catalogue/genesi/Etna_SAR/rdf/?start=2007-06
3. Submit (refined) dataset query
4. Result metadata file
   also provides the URL of the dataset itself
Notes for the developer – Example of queries

- **OpenSearch queries**

- **SAR OpenSearch queries**

- **GPS OpenSearch queries**
A machine to machine example

The following steps are executed via script:

- search the same data used in the portal (syntactic)
- download them locally
- plots a graph from the GPS text file
- creates a kmz file containing the downloaded tiff file and the generated plot
- Open the two kmz files using Google Earth
Notes for the developer – technologies/products

- **Semantic Framework (editor)**
  - Jena
  - Protegè
  - Redland
  - RDFSuite
  - ARQ
  - RDF2GO
  - Semantic Web Client

- **Reasoner**
  - Pellet
  - RacerPRO
  - KAON2
  - OWLIM
  - OntoBroker

- **SPARQL Endpoint**
  - Openlink Virtuoso
  - D2RQ Server
  - Semantic Discovery System
  - Sesame

- **Storage (triplestore)**
  - Sesame
  - Virtuoso
  - AllegroGraph
  - YARS
Notes for the developer – Modelling with Protege

- Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge based applications with ontologies.

- The is-a objects relationship comes from the objects domain hierarchy

- New relationship should be created by means of specific operation

- Different kind of relationship should be used: direct, inverse, functional, transitive etc

- The full Ontology graph with all the relationship could be created by the Ontograph tool (plug-in of Protégé)
Notes for the developer – Ontology graph
ENVRI Ontology has been consolidated with GEMET thesaurus integration

Two different navigation path implemented to reach the GEMET concepts has been implemented
- Theme
- Groups and SuperGroup

Categories rdf description files to integrate GEMET has been used from the Eionet (www.eionet.europa.eu/gemet/)
Semantic SESAME latest framework has been installed in a new powerful server machine with a public IP.
Notes for the developer – SESAME interfaces

- SPARQL Endpoint
- OPENRDF Workbench
- JAVA API
- REST Services
- Remote HW

KNOWLEDGE BASE (SESAME)

ENVRI

ENVRI Web-Services
Resource Harvester: is the component devoted to harvest the metadata from a list of resources in order to initially populate the triple store mapping different data source formats with the needed triples.

The creation of the knowledge base contents will be performed in different phases both as offline or runtime activities. The following schema will help to identify the phases:
- Domain Harvesting at start-up (offline)
- Domain Harvesting update (runtime)
- Users tags provisioning (runtime)

Use of a subset of metadata-model fields content to match ontology terms and tag the selected resource
Notes for the developer – ENVRI DISCOVERY

- Discovery operation involves data retrieving by querying the knowledge base. Being in RDF format, the knowledge base needs to be queried with specific languages. One of the most used is SPARQL.

- The discovery process allows the user to select a set of concepts from the Ontology application domain searching through the knowledge base.

- The semantic discovered results will be shown in the same way as the syntactic discovery operation, allowing the user consumer to find more results in respect to the syntactic search.
Notes for the developer – TAGGING

**AUTOMATIC TAGGING**

1. GENESI_DEC Catalogue
2. GENESI_DEC Core
3. HARVESTER
4. TAGGING Processor
5. GENESI_DEC Knowledgbase

**GENESI_DEC Ontology**

- SBA Thesaurus
- GCMD Thesaurus
- GEMET Thesaurus

**MANUAL TAGGING**

1. GENESI_DEC Catalogue
2. DISCOVERY
3. Terms Selection
4. TAGGING Processor
5. GENESI_DEC Knowledgbase

**LEGENDA**

1. Harvest GENESI-Catalogue both DATASETS and Services
2. Parse of specific resources information (like subject, abstract, etc) to match GENESI-DEC Thesaurus terms
3. If point 2 is positive a TAG for the identified term and resource will be add to the knowledge base by means of Tagging processor
4. A tag with the identified terms will be created in the knowledge base
5. The tag and the resource will be linked in the knowledge base
THANKS

Thanks!
Semantic Discovery – what’s behind: high level architecture
Notes for the developer – SESAME JAVA API

```java
private String[] getNarrowedTermsList(String term) {
    org.openrdf.repository.RepositoryConnection con = null;
    Vector<String> vector = new Vector<String>();
    boolean notEmpty = false;
    try {
        Repository myRepository = new HTTPRepository("SESAME_SERVER, REPOSITORY_ID");
        myRepository.initialize();
        con = myRepository.getConnection();
        String sparqlQuery = "null;"
        boolean flag = false;
        boolean noThird = false;
        sparqlQuery = "SELECT_NARROWED_CONCEPT_FROM_GMET"
        flag = true;
        term = "<" + term + ">");
        System.out.println("PRIMA SPARQL QUERY
" + sparqlQuery);
        sparqlQuery = sparqlQuery.replace("TERMS", term);
        System.out.println("DOPO SPARQL QUERY
" + sparqlQuery);
        try
        {
            TupleQuery tupleQuery = con.prepareTupleQuery(QueryLanguage.SPARQL, tupleQuery.setInfered(true);
            TupleQueryResult result = tupleQuery.evaluate();
            BindingSet bindingset = null;
            List<String> bindingNames = null;
            String[] first, second, third = null;
            String temp1, temp2, temp3 = null;
            int i = 0;
            while (result.hasNext()) {
                notEmpty = true;
            }
        }
    }
}
```
Notes for the developer – SPARQL Query

Query Repository

Query Language: SPARQL

```sparql
PREFIX ESARegistry:<http://www.semanticweb.org/ontologies/2012/12/ESARegistry.owl#>
PREFIX ESARegistry:<http://www.semanticweb.org/ontologies/2012/12/ESARegistry.owl#>
PREFIX Ontology_GENESI_DEC:<http://www.semanticweb.org/ontologies/2010/8/Ontology_GENESI_DEC.owl#>
PREFIX rdfg-1:<http://www.w3.org/2004/03/trix/rdfg-1/>
PREFIX timezone:<http://www.w3.org/2006/timezone#>
PREFIX ns2:<http://www.w3.org/2001/sw-wotab-status/ns#>
PREFIX contact:<http://www.w3.org/2000/10/swap/pim/contact#>
PREFIX wgs84_pos:<http://www.w3.org/2003/10/gps/gps84_pos#>

ORDER BY ?lab
```

Limit results: 100
Include inferred statements

Execute
### Query Result (100)

Limit results: **100**  
The results shown may be truncated.

<table>
<thead>
<tr>
<th>S</th>
<th>Lbl</th>
<th>Def</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEMET:243</td>
<td>&quot;AIDS&quot; @en-gb</td>
<td>The acquired immunodeficiency syndrome is caused by HIV–virus manifested by opportunistic infections and/or malignancies, and the mortality rate is very high. The syndrome results from a breakdown of the body's disease-fighting mechanism that leaves it defenseless against infections, such as pulmonary tuberculosis, Pneumocystis pneumonia, certain blood infections, candidiasis, invasive cervical cancer, Kaposi sarcoma or any of over 20 other indicator diseases. No effective treatment is available. A striking feature of AIDS is the wide spectrum and frequency of infections with life-threatening pathogens seldom seen in normal hosts. The illness may begin with insidious signs and symptoms, and the process may be more diffuse than when the same conditions are seen in other immune-compromised patients. Four patterns of disease occur in AIDS patients. The pulmonary pattern, the central nervous system pattern, the gastrointestinal pattern, and the pattern of fever of unknown origin. Most patients who recover from a given opportunistic infection subsequently either have a relapse or develop a new type of infection. Many patients continue to have a wasting syndrome and experience such infections as oral thrush. Feelings of depression and isolation are common among AIDS patients and can be intensified if health care workers display fear of the syndrome. (Source: WHO) @en-gb</td>
</tr>
<tr>
<td>GEMET:486</td>
<td>&quot;AOX value&quot; @en-gb</td>
<td>Organic halogens subject to absorption. This is a measure of the amount of chlorine (and other halogens) combined with organic compounds. (Source: PORD) @en-gb</td>
</tr>
<tr>
<td>GEMET:505</td>
<td>&quot;ASEAN&quot; @en-gb</td>
<td>Association of Southeast Asian Nations. (Source: MIBI) @en-gb</td>
</tr>
<tr>
<td>GEMET:170</td>
<td>&quot;Africa&quot; @en-gb</td>
<td>The second largest of the continents, on the Mediterranean in the north, the Atlantic in the west, and the Red Sea, Gulf of Aden, and Indian Ocean in the east. The Sahara desert divides the continent unequally into North Africa and Africa south of Sahara. The largest lake is Lake Victoria and the chief rivers are the Nile, Niger, Congo, and Zambezi. The hottest continent, Africa has vast mineral resources, many of which are still undeveloped. (Source: CED / AIRHER) @en-gb</td>
</tr>
<tr>
<td>GEMET:367</td>
<td>&quot;Americas&quot; @en-gb</td>
<td>The landmasses and islands of North America, South America, Mexico, and Central America included in the Western Hemisphere. (Source: AIRHER) @en-gb</td>
</tr>
<tr>
<td>GEMET:370</td>
<td>&quot;Ames test&quot; @en-gb</td>
<td>A bioassay developed by Bruce N. Ames in 1974, performed on bacteria to assess the capability of environmental chemicals to cause mutations. (Source: BJTUL / KOREA) @en-gb</td>
</tr>
<tr>
<td>GEMET:464</td>
<td>&quot;Antarctic&quot; @en-gb</td>
<td>The waters, including ice shelves, that surround the continent of Antarctica, which comprise the southernmost parts of the Pacific, Atlantic, and Indian oceans and also the Four Shoals, Balleny Channel, and Weddell Sea. (Source: BVN / OA) @en-gb</td>
</tr>
</tbody>
</table>
Notes for the developer – Web Services

- Migration of the webservices from SOAP (Simple Object Access Protocol) to REST (REpresentational State Transfer) protocol.

- **http://earth2.cs.telespazio.it/rest/GenesiSemanticService.wsdl** is the endpoint to retrieve the wsdl

- **http://earth2.cs.telespazio.it/rest/GenesiSemanticService/getSemanticTerms** get semantic root elements

- **http://earth2.cs.telespazio.it/rest/GenesiSemanticService/getDetailedSemanticTerms?args0=XXX** get detailed info of XXX element

- **http://earth2.cs.telespazio.it/rest/GenesiSemanticService/discoveryResourceByTerms?args0=SERIESTERM;http://gcmdservices.gsfc.nasa.gov/kms/concept/372b4016-80ab-4126-b6d1-e847bbf0b44f** discover a resource tagged with the selected term

- **http://earth2.cs.telespazio.it/rest/GenesiSemanticService/addTag?args0=dec:genesidec_user:SERIES:seriesURI&args1=dec:average** to tag a resource
## Notes for the developer – Discovery Mapping

<table>
<thead>
<tr>
<th>TYPE</th>
<th>TERM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERIES</td>
<td>Service URI</td>
<td>To retrieve all dataseries used by a selected service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID list</td>
</tr>
<tr>
<td>SERVICES</td>
<td>Dataseries URI</td>
<td>To retrieve all services that uses the selected dataseries Returns resourceID list</td>
</tr>
<tr>
<td>RESOURCETAG</td>
<td>None</td>
<td>To discover all Resources with tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID list</td>
</tr>
<tr>
<td>SERIESTAG</td>
<td>None</td>
<td>To discover all Dataserries with tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID list</td>
</tr>
<tr>
<td>SERVICETAG</td>
<td>None</td>
<td>To discover all Services with tags</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID list</td>
</tr>
<tr>
<td>RESOURCETERM</td>
<td>Ontology TERM</td>
<td>To discover all resources tagged with the selected term. Returns resourceID list</td>
</tr>
<tr>
<td>SERIESTERM</td>
<td>Ontology TERM</td>
<td>To discover all Dataserries tagged with the selected term. Returns resourceID list</td>
</tr>
<tr>
<td>SERVICETERM</td>
<td>Ontology TERM</td>
<td>To discover all Services tagged with the selected term</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID list</td>
</tr>
<tr>
<td>SERIESDETAILTAG</td>
<td>seriesID</td>
<td>To discover all tags of a selected dataseries</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns (TermsID;TermsLabel;TermsDescription) List</td>
</tr>
<tr>
<td>SERVICEDETAILTAG</td>
<td>serviceID</td>
<td>To discover all tags of a selected service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns (TermsID;TermsLabel;TermsDescription) List</td>
</tr>
<tr>
<td>USERTAG</td>
<td>userID</td>
<td>To discover all tags performed by a specified user</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Returns resourceID;Tag</td>
</tr>
</tbody>
</table>
try {

  toJson += taggingText + seriesTitle + seriesAbstract + seriesSubject;
  taggingText = taggingText.toLowerCase();

  // add triple of resource
  iter = res.listProperties();
  int counter = 0;
  while (iter.hasNext()) {
    stmt = (Statement) iter.next();
    if (stmt.getPredicate().toString().equals(xmlns_dc_identifier)) {
      con = myRepository.getConnection();
      org.openrdf.model.Statement statement1 = vf.createStatement(resourceURI, RDF.TYPE, (URI) getURI(catalogue.DATASERIES));
      org.openrdf.model.Statement statementTitle = vf.createStatement(resourceURI, (URI) getURI(catalogue.TITLE), vf.createLiteral(seriesTitle));
      DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ");
      String timestamp = Long.toString(System.currentTimeMillis());
      Date currentDate = new Date(System.currentTimeMillis());
      // Converting date to ISO8601
      String javaDate = dateFormat.format(currentDate);
      String isoDate = javaDate.substring(0, 22) + "" + javaDate.substring(22);
      URI dateResource = vf.createURI(statement1.getSubject() + "_Time");
      org.openrdf.model.Statement statement5 = vf.createStatement(dateResource, RDF.TYPE, (vf.createURI(ontns_time_instant)));
      org.openrdf.model.Statement statement6 = vf.createStatement(dateResource, vf.createURI(ontns_time_xsdTime), vf.createLiteral(isoDate));
      org.openrdf.model.Statement statement7 = vf.createStatement(statement1.getSubject(),
          vf.createURI(ontns_time_life_duration), vf.createLiteral("1");
      con.add(statement1);
      con.add(statementTitle);
      con.add(statement5);
      con.add(statement6);
      con.add(statement7);
      con.close();
    }
  }
}
private String[] discoveryResources(String sparqlQuery) {
    org.openrdf.repository.RepositoryConnection con = null;
    Vector<String> vector = new Vector<String>();
    boolean notEmpty = false;
    try {
        Repository myRepository = new HTTPRepository("SESAME_SERVER", "REPOSITORY_ID");
        myRepository.initialize();
        con = myRepository.getConnection();
        boolean flag = false;
        boolean noThird = false;
        flag = true;
        try {
            TupleQuery tupleQuery = con.prepareTupleQuery(QueryLanguage.SPARQL,
                getRepositoryNamespaces("SESAME_SERVER", "REPOSITORY_ID") + sparqlQuery);
            tupleQuery.setIncludeInferred(true);
            TupleQueryResult result = tupleQuery.evaluate();
            BindingSet bindingSet = null;
            List<String> bindingNames = null;
            String[] first, second, third = null;
            String temp1, temp2, temp3 = null;
            int i = 0;
            while (result.hasNext()) {
                notEmpty = true;
                bindingNames = result.getBindingNames();
                bindingSet = result.next();
                temp1 = bindingSet.getValue(bindingNames.get(0)).toString();
                temp2 = bindingSet.getValue(bindingNames.get(1)).toString();
                vector.add(temp1 + ";" + temp2);
                i++;
            }
            con.close();
        } catch (MalformedQueryException ex) {
            // Handle exception
        }
    } catch (SQLException e) {
        // Handle SQL exception
    }
}
Notes for the developer – Tagging

```java
private void addTermsTag(String user, String type, String resource, String category) {
    org.openrdf.repository.RepositoryConnection con = null;
    ValueFactory vf = null;
    try {
        Repository myRepository = new RDFFactory().getRepository("SESSION_SERVER", "REPOSITORY_ID");
        myRepository.initialize();
        vf = myRepository.getValueFactory();
        java.util.Date today = new java.util.Date();
        String randomString = "_" + new java.sql.Timestamp(today.getTime()).toString();
        URI annotation = vf.createURI(randomString);
        org.openrdf.model.Statement statement1 = vf.createStatement(Annotation,
                                        RDF.TYPE, (vf.createURI(type)));
        // Link the dataset annotation with a Category found
        org.openrdf.model.Statement statement2 = vf.createStatement(Annotation,
                                        vf.createURI("ontds_datasetHasCategory"), vf.createURI(category));
        // Link the Resource with the created dataset annotation
        org.openrdf.model.Statement statement3 = vf.createStatement(vf.createURI(resource),
                                        vf.createURI("ontds_datasetHasAnnotation"), Annotation);
        // User tag
        org.openrdf.model.Statement statement4 = vf.createStatement(Annotation,
                                        vf.createURI("ontds_datasetHasCreator"), vf.createURI(user));
        // Time tag
        org.openrdf.model.Statement statement5 = vf.createStatement(dateAnnotation,
                                        RDF.TYPE, (vf.createURI("ontds_datasetHasDate")));
        org.openrdf.model.Statement statement6 = vf.createStatement(dateAnnotation,
                                        vf.createURI("ontds_datasetHasDateCreated"),InstantLiteral(isoDate));
        org.openrdf.model.Statement statement7 = vf.createStatement(Annotation,
                                        vf.createURI("ontds_datasetHasDateCreated"), dateAnnotation);
        con = myRepository.getConnection();
        con.add(statement1); con.add(statement2);
        con.add(statement3);
        con.add(statement4);
        con.add(statement5);
        con.add(statement6);
        con.add(statement7);
        con.close();
    }
```
Semantic direct access

- [http://earth2.cs.telespazio.it/rest/GenesiSemanticService](http://earth2.cs.telespazio.it/rest/GenesiSemanticService)

**Sparql Queries**

- SELECT DISTINCT ?lab WHERE {{?s rdf:type dec:USABLE_TERMS .}UNION{?s rdfs:subClassOf dec:USABLE_TERMS .}{?s rdfs:label ?lab .}} ORDER BY ?lab