EMSO implementation and operation: DEVelopment of instrument module

DATA MANAGEMENT ARCHITECTURE DESIGN

D6.2

Document identifier:	**EMSODEV-D6.2_V1.3**
Due Date of Delivery to EC	**M12 (August 2016)**
Dissemination level	**PUBLIC**
Actual Date of Delivery to EC	**23/10/2016**
Document date:	**30/06/2016**
Deliverable Title:	**Data Management Platform Architecture Design**
Work package:	**WP6:Data management platform and related services**
Lead Beneficiary:	**INGV**
Other Beneficiaries	**IFREMER, HCMR, CSIC, NERC, MI, UniHB, IPMA, GeoEcoMar, ENG**
Authors:	**Marco Pappalardo, Pasquale Andriani, Lucio Badiali, Raul Bardaji, Rafael Bartolomé, Juanjo Dañobeitia, Oscar García, Alex Hardisty, Robert Huber, Leandro Lombardo, Massimiliano Nigrelli, Dario Pellegrino, Jaume Piera, Markus Stocker**
Document status:	**Final**
Document link:	**https://emdesk.eu/shared/580dbaa340c7c-6f4cf028ffb3329204047319c7981459**

History of changes

Version	Date	Change	Authors
0.1	30.05.2016	ToC completed. First issue of document.	M. Pappalardo
0.2	15.06.2016	Revised ToC. Inserted Sections 1 and 2.	M. Pappalardo
0.3	28.06.2016	Inserted Section 3.	M. Pappalardo
0.4	31.08.2016	Added Sections 4, 5, and Annex A.	P. Andriani , L. Lombardo, M. Nigrelli, D. Pellegrino
0.5	07.09.2016	Added Sections 6 and 7.	R. Bardaji, Rafael Bartolomé, J. Dañobeitia, Oscar García, Alex Hardisty, Robert Huber, Jaume Piera, Markus Stocker.
0.6	09.09.2016	Finalized Section 2. Added Section 8. Full revision.	M. Pappalardo
0.7	16.09.2016	Internal Review. New version after proposed changes integration.	M. Pappalardo
1.0	22.09.2016	First issue released.	M. Pappalardo
1.1	03.10.2016	Executive Board Review	
1.2	06.10.2016	Editing and Proof Reading	Mairi Best
1.3	20.10.2016	Integrating comments from ENVRI+ community	M. Pappalardo

TABLE OF CONTENTS

LIST OF FIGURES

1. EXECUTIVE SUMMARY

This document is the EMSODEV Data Management Platform Architecture Design D6.2 Deliverable describing the design concept underlying all features and services provided by the Data Management Platform for data acquired, processed, archived and distributed during the whole project lifecycle. This document is a formal deliverable for the European Commission, applicable to all members of the EMSODEV project, beneficiaries and Joint Research Unit members.

2. INTRODUCTION

As stated in the contract, D6.2 addresses Task WP6.2 Data management platform architecture design.

This document provides the definition of the data management platform architecture design according to the ENVRI Reference Model. The architecture will integrate regional EMSO nodes' data infrastructures based on international standards.

The data management platform will enable easy and user friendly access to EMSO data and provide data processing, transformation, visualization and analysis functionalities. These functionalities will be integrated within the architecture through a set of common services, in compliance to the ENVRI Reference Model.

The architecture includes the definition of all the functionalities that the platform will offer, the interconnections between all the modules that compose the architecture, the stakeholder map and their roles in the whole data management process.

This report is structured as follows:

1. **Section 2** contains this introduction to the Deliverable.
2. **Section 3** provides an overview of the ENVRI Reference Model.
3. **Section 4** analyzes the aspects of EMSODEV scenarios related to Data Management Platform.
4. **Section 5** depicts the Architectural Design of the EMSODEV Data Management Platform.
5. **Section 6** describes the Data Management Platform Tools.
6. **Section 7** discusses the interoperability issues with EMSO Regional Nodes.
7. **Section 8** contains the final considerations.

At the end of this Report also an Annex is provided (Annex A). This reads the detailed Application Programming Interface (API) for the EMSODEV Data Management Platform.

3. THE ENVRI REFERENCE MODEL

3.1. Overview

The objectives of ENVRI were born from examining the design of the six ESFRI environmental Research Infrastructures (ENV RIs), (namely EMSO, ICOS, EURO-Argo, EISCAT-3D, LifeWatch, and EPOS) in order to identify common computational characteristics and develop an understanding of specific requirements through observations.

Throughout the study, a standard model, Open Distributed Processing (ODP), is chosen to interpret the design of the research infrastructures, and place their requirements into the ODP framework for analysing. The document reports the initial results from this study. Briefly, from the aspect of the ODP Engineering Viewpoint, the architectural characteristics of the RIs have been examined, and five common sub-systems have been identified: data acquisition, curation, access, processing and community support. Secondly, from the aspect of the ODP Computational Viewpoint, we looked at each of the six RIs in detail and identified the common functions and embedded computations they provided. Matrices have been used for comparison. Definitions of functionalities have been provided. Finally, from the aspect of the ODP enterprise viewpoint, four common communities were identified and the community roles derived.

The contribution of this work to environmental science research infrastructures is threefold:

- it investigates 6 ESFRI RIs, which are a collection of representative research infrastructures for environmental sciences, and provides a projection of Europe-wide requirements they have; identifying in particular, those they have in common;
- it experiments with ODP as an approach in requirement analysis, to serve as a common language for interpretation and discussion to ensure unifying understanding;
- the results from this study can be used as an input to a design or implementation model. Common services can be provided in light of common analysis, which can be widely applicable to various environmental research infrastructures and beyond.

Starting from the identification of the common requirements of the ENV RIs, the ENVRI RM finally reached the goal to produce a common model for them. Throughout the study, ODP had been used as the analysis framework, which serves as a uniform platform for interpretation and discussion to ensure a unified understanding.

The benefit of using the ENVRI RM (inherited from ODP) to analyse the requirements for environmental research infrastructures is also threefold:

- to use a standardised language to interpret the design of research infrastructures which helps to unify understanding;
- to provide a gentler pathway to link real-world systems to an ODP model world;
- since the ODP framework is created to help designers deliver a practical architecture which leads to concrete implementations, using ODP concepts for requirements analysis can help us to drill down to details and identify essential problems.

3.2. Open Distributed Processing

Open Distributed Processing (ODP) is a framework for specifying systems, whether distributed or not. It enables implementation of a reference model (RM-ODP) in computer science, which provides a coordinating framework for the standardization of open distributed processing (ODP). It supports distribution, interworking, platform and technology independence, and portability, together with an enterprise architecture framework for the specification of ODP systems.

The RM-ODP view model provide five generic and complementary viewpoints on the system and its environment. RM-ODP, also named ITU-T Rec. X.901-X.904 and ISO/IEC 10746, is a joint effort by the International Organization for Standardization (ISO), the International Electrotechnical Commission (IEC) and the Telecommunication Standardization Sector (ITU-T).

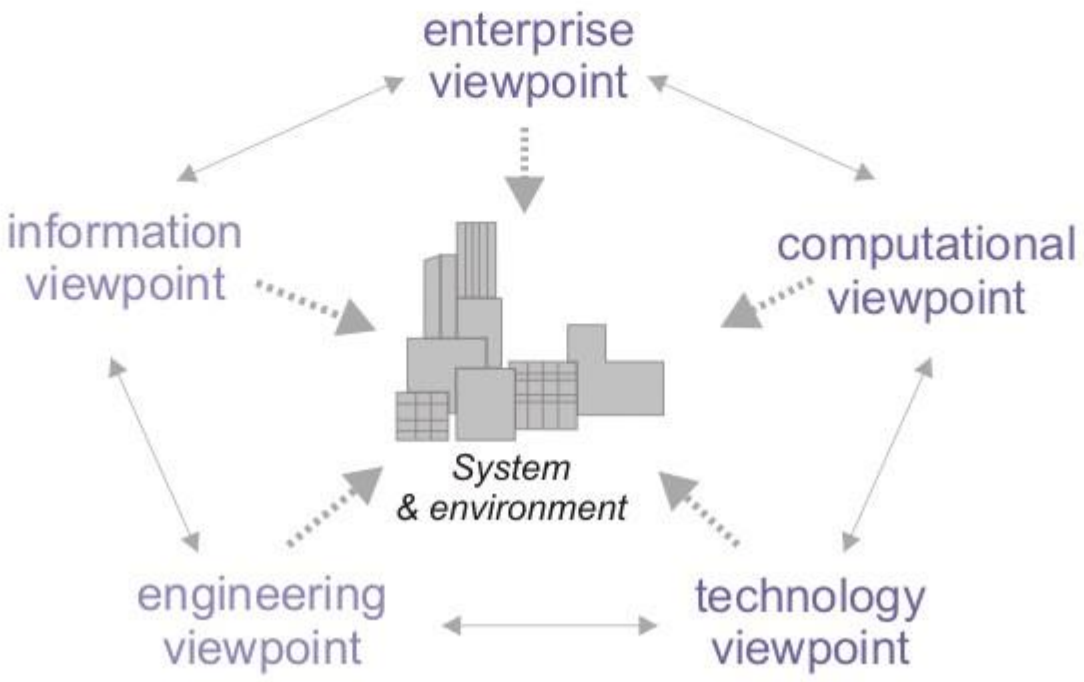


Figure 1 - The RM-ODP view model.

The RM-ODP is a reference model based on precise concepts derived from current distributed processing developments and, as far as possible, on the use of formal description techniques for specification of the architecture. Many RM-ODP concepts, possibly under different names, have been around for a long time and have been rigorously described and explained in exact philosophy (for example, in the works of Mario Bunge [R27]) and in systems thinking (for example, in the works of Friedrich Hayek [R28][R29]). Some of these concepts — such as abstraction, composition, and emergence — have recently been provided with a solid mathematical foundation in category theory.

RM-ODP has four fundamental elements:

- an object modelling approach to system specification;
- the specification of a system in terms of separate but interrelated viewpoint specifications;
- the definition of a system infrastructure providing distribution transparencies for system applications; and
- a framework for assessing system conformance.

The RM-ODP family of recommendations and international standards defines a system of interrelated essential concepts necessary to specify open distributed processing systems and provides a well-developed enterprise architecture framework for structuring the specifications for any large-scale systems including software systems. In terms of compliance to standards, RM-ODP consists of a set of basic ITU-T Recommendations and ISO/IEC International Standards.

3.3. From ODP Viewpoints to ENVRI Services

Most complex system specifications are so extensive that no single individual can fully comprehend all aspects of the specifications. Furthermore, we all have different interests in a given system and different reasons for examining the system's specifications. A business executive will ask different questions of a system make-up than would a system implementer. The concept of an RM-ODP viewpoints framework, therefore, is to provide separate viewpoints into the specification of a given complex system. These viewpoints each satisfy an audience with interest in a particular set of aspects of the system. Associated with each viewpoint is a viewpoint language that optimizes the vocabulary and presentation for the audience of that viewpoint.

Viewpoint modelling has become an effective approach for dealing with the inherent complexity of large distributed systems. Current software architectural practices, as described in IEEE 1471, divide the design activity into several areas of concerns, each one focusing on a specific aspect of the system. Examples include the "4+1" view model, the Zachman Framework, TOGAF, DoDAF and, of course, RM-ODP.

A viewpoint is a subdivision of the specification of a complete system, established to bring together those particular pieces of information relevant to some particular area of concern during the analysis or design of the system. Although separately specified, the viewpoints are not completely independent; key items in each are identified as related to items in the other viewpoints. Moreover, each viewpoint substantially uses the same foundational concepts (defined in Part 2 of RM-ODP). However, the viewpoints are sufficiently independent to simplify reasoning about the complete specification. The mutual consistency among the viewpoints is ensured by the architecture defined by RM-ODP, and the use of a common object model provides the glue that binds them all together.

More specifically, the RM-ODP framework provides five generic and complementary viewpoints on the system and its environment:

I. The enterprise viewpoint, which focuses on the purpose, scope and policies for the system. It describes the business requirements and how to meet them.
II. The information viewpoint, which focuses on the semantics of the information and the information processing performed. It describes the information managed by the system and the structure and content type of the supporting data.
III. The computational viewpoint, which enables distribution through functional decomposition on the system into objects which interact at interfaces. It describes the functionality provided by the system and its functional decomposition.
IV. The engineering viewpoint, which focuses on the mechanisms and functions required to support distributed interactions between objects in the system. It describes the distribution of processing performed by the system to manage the information and provide the functionality.
V. The technology viewpoint, which focuses on the choice of technology of the system. It describes the technologies chosen to provide the processing, functionality and presentation of information.

In ODP, the purpose of the Engineering Viewpoint is to identify and specify the structuring mechanisms for distributed interactions and the functional elements. It concerns the architectural features of an infrastructure.

The structures of the studied RIs can be divided into sub-systems based on functions and locations of computational elements. For the purposes of this document, each sub-system is defined as a set of capabilities that collectively are defined by a set of interfaces with corresponding operations that can be invoked by other sub-systems. An interface in ODP is an abstraction of the behaviour of an object that consists of a subset of the interactions of that object together with a set of constraints on when they may occur. Sub-systems are disjoint from each other. Five common sub-systems are identified: data acquisition, data curation, data access, data processing, and community support. The order of these sub-systems is irrelevant.

The data acquisition sub-system collects raw data from sensor arrays, various instruments, or human observers, and brings the measurement and observation data streams into the system. Note, ENVRI is concerned with the computational aspects of an infrastructure, thus, by definition and for many RIs, the data acquisition sub-system starts from raw sensor signals being converted into digital values and received by the system. It is also true that other RIs can also acquire processed data, or data products, as their "raw data" and further process these. This is the case of LifeWatch. This is also one of the motivators for the life-cycle introduced in ENVRI RM 2.0, where data use feeds back into data acquisition. So, the output of data use in RI X can be the input of data acquisition in RI Y. This output may be data products, processed sensor signals. Anyway, in the first type of RIs there are many activities related to raw data "provisioning" including, defining data acquisition protocols, design and deployment of the sensor instruments, and configuration and calibration devices, which are crucial tasks for data acquisition nevertheless beyond the scope of the ENVRI investigation. The data acquisition sub-system is typically operated at observatories or stations. Data in the acquisition sub-system are normally non-reproducible, the so-called raw data or primary data. Consistent time-stamps are assigned to each data object. There are the cases that the raw data may be generated by a simulation model, in which situation the raw data may be reproducible in terms of being regenerated. The (real-time) data streams sometimes may be temporarily stored (e.g., in computer clusters), and then sampled, filtered or processed (e.g., based on applied quality control criteria). Control software is often provided to allow the execution and monitoring of data flows.

The data collected at the data acquisition sub-system are transmitted to the data curation sub-system, to be maintained and archived there. The data curation sub-system facilitates quality control and preservation of scientific data. It is typically operated at a data centre. Data handled at the curation sub-system are often reproducible, i.e. can be re-processed. Operations such as data quality verification, data identification, annotation, cataloguing, and long-term preservation are often provided. Various data products are generated and provided for users. They, too, need to be accessed through the data access sub-system. There is usually an emphasis on non-functional requirements for a data curation sub-system including the need for satisfying performance criteria in availability, reliability, utility, throughput, responsiveness, security and scalability.

The data access sub-system enables discovery and retrieval of data housed in data resources managed by a data curation sub-system. Data access sub-systems often provide facilities such as data portals, as well as services to present or deliver the data products. Search facilities including both query-based and navigation-based searching tools are provided which allow users or services to discover interesting data products. Discoveries based on metadata or semantic linkages are most common. In the last decade, active studies and developments have resulted in large-scale representative scientific digital libraries and archives created for different knowledge and data domains; these repositories begun to play an important role in supporting scientific studies and in increasing their efficiency. Basic functions of such systems consist in publishing research results or data acquisition campaign outputs and in enabling wide open access to them. In the scenarios we depicted in this document, the functional capabilities of the data repository can be extended by offering users the opportunity of linking information objects and declaring explicitly semantics of created linkages based on a given ontology; we refer to such linkages as semantic linkages. Those are a direct outcome of adoption of the Reference Model.

It is supposed that the users have different motives to create, on their own initiative, binary oriented semantic linkages between information objects of the repository content; it is also supposed that the users can receive notifications about new linkages created or about changes in properties of existing ones. They can also respond to such events by creating new linkages. Participants of semantic linkages may be scientific publications, sets of scientific data, profiles (metadata) of their creators and other users registered in the repository, profiles of organizations in which they work, research programs represented in the form of digital documents, scientific reports, reviews, project descriptions, ontologies of different knowledge domains, software features and their descriptions, specifications of metadata standards, etc.

Data handled at the access sub-system can be either structurally and semantically homogeneous or heterogeneous. When supporting heterogeneous data, different types of data (often pulled from a variety of distributed data resources) may be converted into uniform representations with uniform semantics which can be resolved by a data discovery and access service. Services allowing harvesting of metadata and/or data, as well as services enhancing the performance by compression and packaging methods and encoding services for secure data transfer are often part of the data access sub-system. Data access can be open or controlled (e.g., enforced by authentication and authorisation policies). It is notable that a data access sub-system usually does not provide "write" operations for end users, although such operations may be provided for an administrator of a data resource.

The data processing sub-system aggregates the data from various resources and provides computational capabilities and capacities for conducting data analysis and scientific experiments. Data handled by the data processing sub-system are typically derived and recombined via the data access sub-system. A data processing sub-system normally offers operations for statistical and/or mining functions for analysis, facilities for conducting scientific experiments, modelling/simulation, and scientific visualisation. Performance requirements for processing scientific data tend to be concerned more about scalability issue, which may also be necessary to address at the infrastructure level --for example, to make use of the Grid or Cloud technology. In this case, functionalities to interact with the physical infrastructure should be provided.

Finally, the community support sub-system manages, controls and tracks users' activities and supports users to conduct their roles in communities. Data handled by a community support sub-system typically are user generated data, control and communications. A community support sub-system normally supports interactive visualisations, Authentication, Authorisation and Accounting (AAA), as well as managing virtual organisations. The community support is orthogonal to and cross-cutting the other 4 sub-systems. There may be other ways to group the functional elements,. The main purpose for the classification is to identify the common structural characteristics of the environmental research infrastructures.

3.4. EMSODEV Services Overview

The five sub-systems map well to the architectures of the RIs studied. **EMSO RI** mainly focuses on **data acquisition**, **curation and access**. It is in fact typical of **large-scale observatory systems**.

The ODP **Computational Viewpoint** focuses on the functionality of an infrastructure, and the service it offers. Dividing the structures of RIs into sub-systems helps to break down the complexity in analysis. Within each sub-system, a data-oriented approach was used. It follows the data life-cycle -- e.g. creation, transmission, transformation, modification, processing, and visualisation -- to identify key functions and embedded computations.

EMSO, the European Multidisciplinary Seafloor and water-column Observatory, is a European consortium of sea floor observatories for the long-term monitoring of environmental processes related to ecosystems, climate change and geo-hazards.

The objectives of the EMSO community are to ensure the technological and scientific framework for the investigation of marine environmental processes related to the interaction between the

geosphere, biosphere, and hydrosphere and for sustainable management through long-term monitoring with real-time data transmission.
EMSO nodes will include a common set of sensors for basic measurements and further sensors for specific purposes defined by users. The common set of instruments comprises seismometers, hydrophones for geophysics, magnetometers, gravity meters, CTD (Conductivity, Temperature, and Depth), current meters, chemical sensors, pressure sensors, and hydrophones for bio-acoustic monitoring. This common set of instruments led to the definition of the EMSO Generic Instrumentation Module, aka EGIM. Additionally, laboratory studies are performed on material collected at these sites by sampling devices (e.g., water samplers, sediment cores, traps etc.). The following activities are carried out at individual EMSO nodes. They are likely supported by computational facilities in order to:

- Design measurements and monitoring models based on geographical location, scientific requirements, operational requirements, and available resources;
- Develop and test the sensors;
- Deploy the instruments into selected locations of the deep-sea;
- Adapt the infrastructure to new deployed instruments. The EMSO-ERIC Science, Technology, and Ethics Advisory Board and the Executive Board will establish general and detailed requirements and standards in order to fulfil both cabled node and stand-alone node integration into a unique research infrastructure;
- Recover and reset the sensors;
- Update with new technology e.g., using cabled systems;
- Send data from sensors to surface buoys/boats, to satellites, then forward to shore stations.

EMSO data collected in experiments at eleven regional sites are locally stored and organized in catalogues or relational databases run by the institutions involved. Some of EMSO observatories' data from some distributed sites are harvested and long term archived at three data archives, Ifremer (EUROSITES[1]), UniHB (PANGAEA[2]) and INGV (MOIST[3]). A central archive hosting a web-service access to all the databases is planned for the near future. We consider a group of functions provided by MOIST, PANGEA and EUROSITES that support data quality control and preservation as a ***data curation sub-system***. Key functions include but are not limited to:

- **Data Identification**: e.g., assigning persistent identifiers. EMSO partially already assigns DOIs to some of its data sets, e.g., from the HAUSGARTEN site, and will use DOIs for further data products;
- **Data Cataloguing**: EMSO collects metadata on both the physical sensors and observatories as well as on the data. Observatories are intended to be described by SensorML;
    - Metadata are provided by the regional nodes of EMSO: MOIST[3] is a data management system for multi-parametric observatories, aimed at hosting multidisciplinary data and metadata. The core part is the database that indexes data and keeps track of the data source. MOIST supports 11 EMSO sites in organising, indexing and transforming data into a compatible data scheme. MOIST is developed to adopt the most common standards (e.g., OGC, NASA, INSPIRE) for organising its information system. It offers access to data through NASA DIF and Dublin Core via a OAI-PMH interface as well as an OpenSearch interface. The information system PANGAEA offers a variety of services for scientific

[1] EUROSITES: http://www.eurosites.info/about.php
[2] PANGAEA: http://www.pangaea.de/
[3] MOIST: http://moist.rm.ingv.it/

project data management, long-term data archiving and data publication. PANGAEA provides access to metadata in several formats such as DC, NASA DIF, ISO19139 or DarwinCore. Access to metadata is provided via a OAI-PMH interface, OpenSearch as well as DiGIR. Ifremer offers access to several EMSO sites via their EUROSITES data management system, which offers access to data in NetCDF format via FTP.

  - EMSO has implemented a prototype common data catalogue[4] which uses the OpenSource panFMP[5] software to harvest and index the metadata records. PANGAEA and MOIST metadata are harvested via their OAI-PMH interfaces while for Ifremer an additional service has been implemented which extracts metadata from the NetCDF records which then are harvested via http. The EMSO common data catalogue offers a common OpenSearch interface as well as a metadata transformation service which offers metadata in dclite4g format compliant with the GENESI requirements used for the ENVRI OpenSearch client;

- **Data Preservation**: e.g., in PANGAEA, a curator is responsible for the data archiving and publication[6]. MOIST will adopt a reference model developed inside SCIDIP-ES EC project.

The PANGAEA data library and publisher retrieves data from EMSO resources and make them publically accessible. We consider a group of functions that facilitates the publication and access of EMSO data as a ***data access sub-system***. This sub-system includes the following functions:

- **Data Conversion**. PANGAEA provides the following tools/software:
  - Pan2Applic[7], which converts files or folders of files (ascii/tab-separated data files with or without metaheader), downloaded from PANGAEA via the search engine or the data warehouse to formats as used by applications, e.g. for visualization or further processing;
  - PanTool[8] which is used for data conversion and recalculation, written to harmonize individual data collections to a standard import format used by PANGAEA;
  - Split2Events[9], which splits one file with data from several events into several files, one for each event;
  - PANGAEA and MOIST are planning to provide NetCDF transformation services.
- **Data Visualisation**. PANGAEA provides the following tools/software:
  - PanPlot[10], which allows the visualisation of data versus time or space in standard x-y-plots or ternary diagrams;
  - PanMap[11], which is for the geographical presentation of data in maps;
  - GIS[12]. Visualisation of geo-referenced data in PANGAEA through GIS (Geographical Information System) functionality enabled by using Google Earth;

MOIST provides the following tools/software:

  - MOIST Plot, a multi-parametric plot in an interactive area of an online web page, for a first immediate data analysis by the user, before download of selected data.

[4] EMSO common data catalogue: http://dataportals.pangaea.de/emso

[5] panFMP: www.panfmp.org

[6] PANGAEA data curation and management: http://wiki.pangaea.de/wiki/Project_data_management

[7] PANGAEA Pan2Applic: http://wiki.pangaea.de/wiki/Pan2Applic

[8] PANGAEA PanTool: http://wiki.pangaea.de/wiki/PanTool

[9] PANGAEA Split2Events: http://wiki.pangaea.de/wiki/Split2Events

[10] PANGAEA PanPlot: http://wiki.pangaea.de/wiki/PanPlot

[11] PANGAEA PanMap: http://wiki.pangaea.de/wiki/PanMap

[12] PANGAEA GIS: http://wiki.pangaea.de/wiki/GIS

- **Data Publication**: PANGAEA offers a DOI resolution service and makes internal use of a DOI registration service to register the DOI metadata records at DataCite. In cooperation with publishers such as Elsevier PANGAEA provides a cross linking service which allows 'reciprocal linking' – automatically linking research data sets deposited in PANGAEA to corresponding articles in Elsevier journals on its electronic platform ScienceDirect and vice versa;
- **Data Import**: Data import tools[13] are provided by PANGAEA;
- **Data Discovery**: a Google-like advanced metadata discovery is provided for public access[14]; also the Common EMSO data catalogue and data portal;
- **Data Citation**: PANGAEA supports data citation[15]. Each data publication is fully citable with a DOI, and can be cross-referenced with journal articles. It also supports pre-publication curatorial review processes;
- **Quality Verification**: PANGAEA's data policy[16] describes the quality assurance for data submission. QC procedures are maintained within the PANGAEA data curatorial process[17] during which quality flags can be assigned to indicate the quality of each measurement;
- **Metadata Harvesting**:
  - PANGAEA OAI-PMH for ESONET data from EMSO sites: harvesting test, integration into ENVRI metadata catalogue etc.;
  - PANGAEA GeoRSS : Embedding GeoRSS feed;
  - Ifremer SOS (Sensor Observation Service) for EUROSITES oceanographic data from EMSO sites: getCapabilities, getObservation, check O&M format;
  - PANGAEA SOS for INGV data from EMSO sites (via MOIST: moist.rm.ingv.it): getCapabilities, getObservation, check O&M format;
  - MOIST OpenSearch for INGV data and metadata from EMSO sites: Data and metadata search according to time or space or parameter;
  - Common NetCDF metadata extraction and transformation service;
  - MOIST OAI-PMH for harvesting INGV data and metadata from EMSO sites.
- **Identifier Registration**: a catalogue of EMSO DOIs is being planned in order to register EMSO DOIs;
- **Metadata Registration**: a centralised Metadata Catalogue to store metadata harvested from distributed sites is implemented within the prototype EMSO data catalogue and data portal (see above);
- **Sensor Registration**: EMSO aims to implement core standards of the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) suite of standards, namely the OGC standards SensorML, Sensor Registry, Catalogue Service for Web (CS-W), Sensor Observation Service (SOS) and Observations and Measurements (O&M). A sensor registry is available at ESONET[18].

We consider a group of functions that support EMSO users to conduct various tasks as a ***community support sub-system***. We identified the following functions:

- **Accounting**: the statistics of the portal accesses is planned to replace the user registration, which can track resource consumption by users for the purpose of capacity and trend analysis;

[13] PANGAEA data import tool: http://wiki.pangaea.de/wiki/Import

[14] PANGAEA advanced metadata discovery: http://www.pangaea.de/

[15] PANGAEA data citation: http://wiki.pangaea.de/wiki/Citation

[16] PANGAEA data policy: http://wiki.pangaea.de/wiki/Data_policy

[17] PANGAEA data quality control: http://wiki.pangaea.de/wiki/Data_policy#Quality_assurance

[18] ESONET: http://vps.dbscale.com:8080/esonet/

- **Metadata Submission**: PANGAEA provides an online metadata & data submission service[19], which supports metadata standards such as ISO19115, Dublin Core, DIF, DarwinCore, and DataCite metadata;
- **Curation Editor**: PANGAEA also provides an online curation editor[20] to be used by curators for the administration of metadata and the import of data;
- **Event Notification**: real-time access of the sensor data to identify new phenomena, and events occurring to provide geo-hazard warning.

EMSO provides advanced technology in data publication and citation through the PANGAEA system. EMSO also offers capabilities for data access, standardisation/harmonisation and visualisation via the MOIST data infrastructure. Three regional sites data are integrated into MOIST, and one regional site is integrated into PANGAEA which additionally offers data from several related or preparatory studies for other EMSO sites. In addition, Ifremer offers access to data from all EUROSITES sites which are shared with EMSO. EMSO has integrated all its operational sites within a common data portal. In the next step, EMSO plans to continue to harmonize its vocabularies and terminologies according to SEADATANET standards and aims to offer access to data via a common NetCDF format which is compliant with SEADATANET. Further EMSO plans are to improve standardised access to real time data via SOS.

From the ***Science Viewpoint***, concerning the organisational and social context, scientific processes, and capturing the purpose, scope and policies of a system, the system itself is represented by one or more enterprise objects within a community, and by the roles in which these objects are involved. Using these concepts, in the following, we identify the common communities in EMSO and their roles.

Following the ENVRI Reference Model we distinguish on the basis of their main objectives and activities:

- **Data Acquisition Community**, who collects raw data and bring (streams of) measures into a system: in EMSO *Designers for measurements and monitoring models*, *Technicians for the development and deployment of the sensors and sensor network*, *Technicians for the operation and maintenance of the sensors and sensor network*, and *Observers/Measurers/Data collectors* collaborate to build one of the most impressive Big Data communities in the world.
- **Data Management Community**, who curates the scientific data, maintains and archives them, produces various data products with metadata, and makes them publically accessible: in EMSO this includes scientists acting as *Storage Managers*, *Curators/Data Managers*, *Data Publishers* and *External Data Providers*.
- **Data Service Provision Community**, who provides various services, applications and software/tools to link, and recombine data and information in order to derive knowledge: in EMSO this involves *Data* and *Data Service Providers* as well as *Other RIs and Networks with overlapping domain interests*. The ENVRI RM does not define terms as Data and Information. That is the reason why EMSODEV Work Package 6 work started from providing a clear definition for these terms within the project community and goal.

[19] PANGAEA online metadata & data submission service: http://www.pangaea.de/submit/
[20] PANGAEA online curation editor: http://wiki.pangaea.de/wiki/4D

- **Data User Community**, who make use of the data and service products: EMSO *Data Users are Internal Scientists/researchers who perform in-house experiments/analyses*, *External Scientists/researchers*, *Technologists/engineers*, *Educators/trainees*, *Police/decision makers*, *Private sector members (Industry investors/consultants)*, *General public/media/citizens (scientists)*.

4. EMSODEV SCENARIO ANALISYS

This section focuses on how the EMSODEV Data Management Platform (as a part of the EMSO Research Infrastructure) maps onto the ENVRI Reference Model v2.0 [R5] (ENVRI-RM) which has been released on July 27, 2016. In particular, the scope of this section is limited to the design of the EMSODEV Data Management Platform (DMP), in terms of services and interfaces, described by using the ENVRI-RM Computational Viewpoint (CV); therefore, other aspects of the EMSO RI (e.g. sensors, EGIM, etc.) are not going to be detailed but only mentioned.

The purpose of using the Computational Viewpoint is to identify a standard set of components and interfaces that the EMSODEV DMP has to address. The model does not specify how these interactions should be implemented – indeed, over the course of the lifetime of a research infrastructure, implementation may change. Nevertheless, the set of the most important interactions should not vary regardless of implementation changes.

In the next sections, where the Computational Viewpoint is applied to the design of the EMSODEV DMP, two primary results are presented: i) the mapping of EMSODEV DMP agents and services to ENVRI-RM computational objects and ii) the definition of the interactions that occur when two or more computational viewpoint objects interfaces are bound together. In particular, the description of the EMSODEV DMP services is presented through the five ENVRI-RM phases (data acquisition, data curation, data publishing, data processing, data use) of the research data lifecycle.

It is worthwhile to note that the pictures included in the next sections are clearly inspired by the ones presented in the ENVRI-RM [R5].

4.1. Data Acquisition Phase

The basis for environmental research to be conducted within the EMSODEV project is the observation and measurement of a wide range of ocean parameters. The data acquisition phase is focused on the mechanisms to be put in place to allow the EMSODEV DMP to harvest data from an extended network of instruments (EGIMs) deployed at sea on observatories located in different European countries.

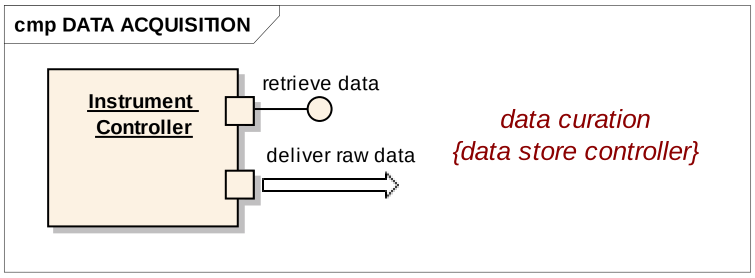


Figure 2 - Data Acquisition

The **data acquisition** components provide pathways by which data sources are integrated into the EMSODEV DMP and deliver data to suitable data stores. Data acquisition is computationally described as a set of **instrument controllers** which encapsulate the accessible functionalities of instruments and other raw data sources out in the field.

The collection of raw scientific data requires coordination between the data acquisition phase (which extracts the raw data from instruments) and the (to be described next) data curation phase (which packages and stores the data).

4.2. Data Curation Phase

After scientific data have been gathered, they must then be collected, catalogued and made accessible to all authorised users.

Bearing this in mind, the **data curation** objects provide the core services required for data preservation and management.

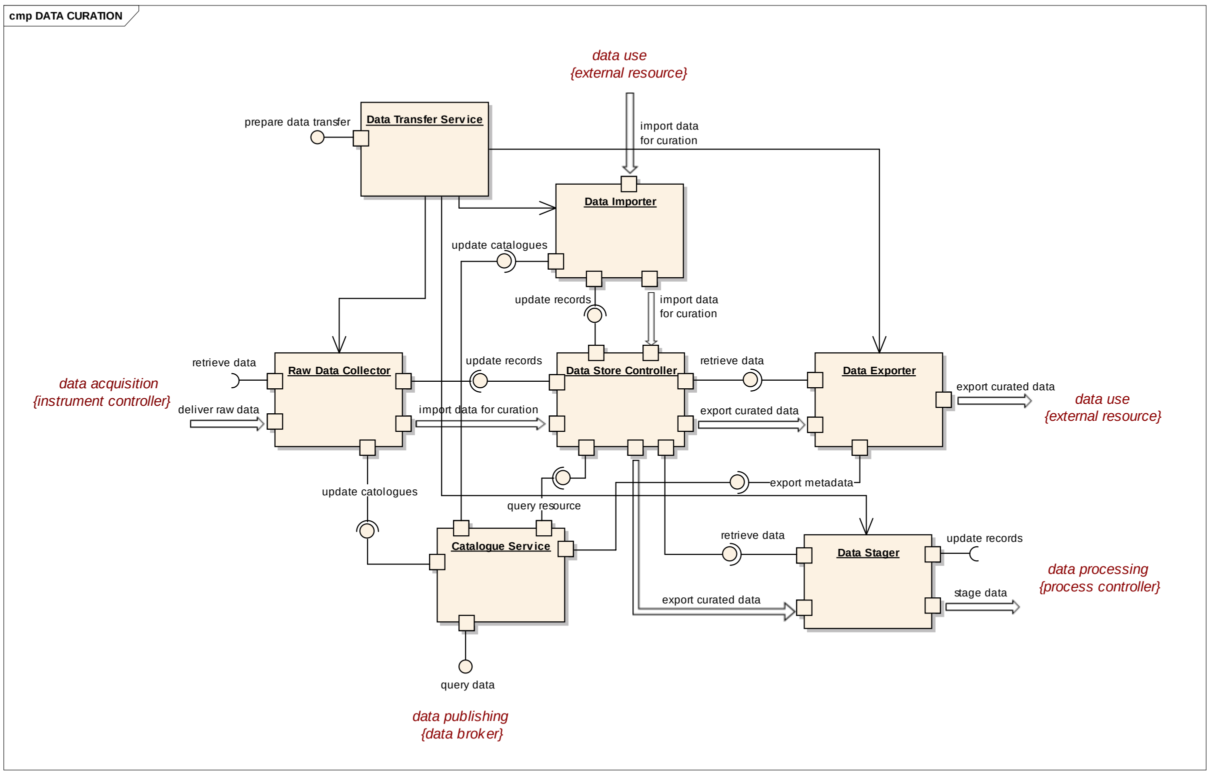


Figure 3 - Data Curation

Computationally, data curation is handled by a set of **data store controllers** (encapsulating the interfaces required to use data stores within the data management platform).

Data store controllers provide access to data stores that may have their own internal data management regimes (NoSQL databases, a distributed file system and a time series database in the context of the EMSODEV DMP). A **data transfer service** is able to provide *data transporters* (such as **raw data collectors**, **data exporters, data importers** or **data stagers**) for managing the movement of data from one part of the data management platform to another.

Each data transporter establishes (multiple, persistent) data channels between instruments (data sources in general) and data stores.

A **raw data collector** can initiate data transfer by requesting data from one or more *instrument controllers* and preparing one or more **data store controllers** to receive the data.

The **raw data collector** is considered responsible for packaging any raw data obtained into a format suitable for curation - this may entail chunking data streams, wrangling data (e.g. change the format from XML to JSON) and associating metadata to the resulting datasets. It also registers any immediately apparent data characteristics in data catalogues - this is done by invoking an update operation on the **catalogue service**.

When curated data needs to be made available to external users/systems, the **data transfer service** will configure and deploy a **data exporter**. This exporter will *retrieve data* from all necessary data stores, opening a data-flow from data store to an external resource; the exporter is also responsible for the repackaging of exported datasets where necessary – this includes the integration of any additional metadata or provenance information stored separately within the data management platform that needs to be packaged with a dataset if it is to be used independently of the platform;

as such, the exporter can invoke the **catalogue service** to retrieve additional meta-information via its *export metadata* interface.

On the other hand when curated data needs to be acquired from external sources/systems, the **data transfer service** will configure and deploy a **data importer**; this importer will open a data-flow from an external resource to one or more suitable data stores within the data management platform and *update records* within those stores as appropriate; the importer is also responsible for registering metadata; as such, the importer can invoke the **catalogue service** to *update catalogues*.

The platform needs also a process to allow subsets of curated data to be "managed" (e.g. moved to a temporary "staging zone") before being further analysed. With this aim, the **data transfer service** will then configure and deploy a **data stager**. The data stager coordinates the transfer of data between curation and processing subsystems.

4.3. Data Publishing Phase

Aside from the curation of scientific data, the EMSODEV DMP provides the means to access that data. Access can be provided in a number of ways, including the export of curated datasets and the querying of data catalogues.

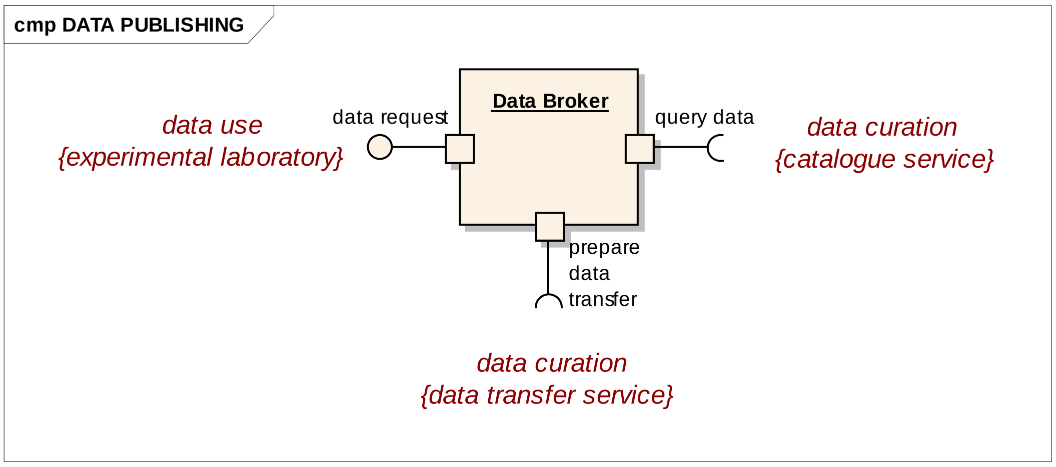


Figure 4 - Data Publishing

The **data publishing** objects provide **data brokers** that act as intermediaries for access to data held within the data store objects supporting data curation.

These brokers are responsible for verifying the agents making access requests and for validating those requests prior to sending them on to the relevant data curation service.

Requests to a data broker in the data publishing subsystem are submitted (typically from an experiment laboratory) via its *data request* interface (for example requests for data to be exported/imported or queried). The data broker will then translate any valid requests into actions.

In the export/import scenario, a data transfer request is then sent to the data transfer service using the *prepare data transfer* interface whilst in the query scenario, the data broker will invoke the catalogue service via its *query data* interface.

4.4. Data Processing Phase

This phase mainly deals with extensive post-processing and analysis of scientific data being curated within the EMSODEV data management platform in order to extract new results of interest for the scientific community.

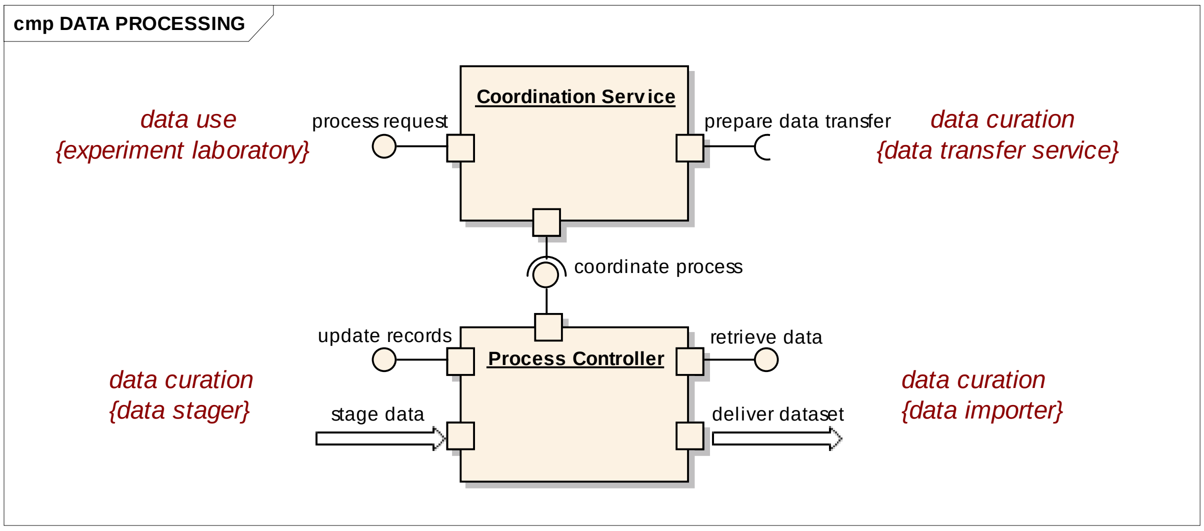


Figure 5 - Data Processing

Data processing objects are described as a set of **process controllers** (representing the computational functionality of registered execution resources) monitored and managed by a **coordination service**. The **coordination service** delegates all processing tasks sent to particular execution resources, coordinates multi-stage workflows and initiates execution.

Requests for data processing (typically originated from an experiment laboratory) are sent to a **coordination service** that will interpret requests and coordinate any internal resources needed accordingly.

In a scenario where a subset of the curated data is going to be used to perform further analysis, it needs to be staged onto an execution platform from a curated data store. In such a case, first the **coordination service** will request that a *data transfer service* within the data curation subsystem *prepare a data transfer*. The data transfer service will then configure and deploy a *data stager*. A data-flow is established between all required *data stores* and all **process controllers** (representing in this case the execution platform on which data processing will be executed) by requesting data via the data stores' *retrieve data* interfaces and *updating records* on the process controllers' side (to ensure that a destination for the data has been prepared and that all data is tracked correctly).

In the case of derived data (being created as a result of the previous analysis) that should be reintegrated into the platform itself, the coordination service will invoke a *data transfer service* via its *prepare data transfer* interface. That data transfer service will configure and deploy a *data importer* to coordinate the transfer of results back into the data curation subsystem. The data importer *retrieves data* from any **process controllers** producing new data and establishes a data-flow from the execution platform underlying those controllers to suitable *data stores* needed to host the data. The *update records* interface is used to prepare data store controllers and to ensure that the ingested data is properly recorded.

4.5.Data Use phase

The modes of interaction between the EMSODEV data management platform and the broader scientific community that it serves are going to be accounted for in the design of the platform within the data use phase.

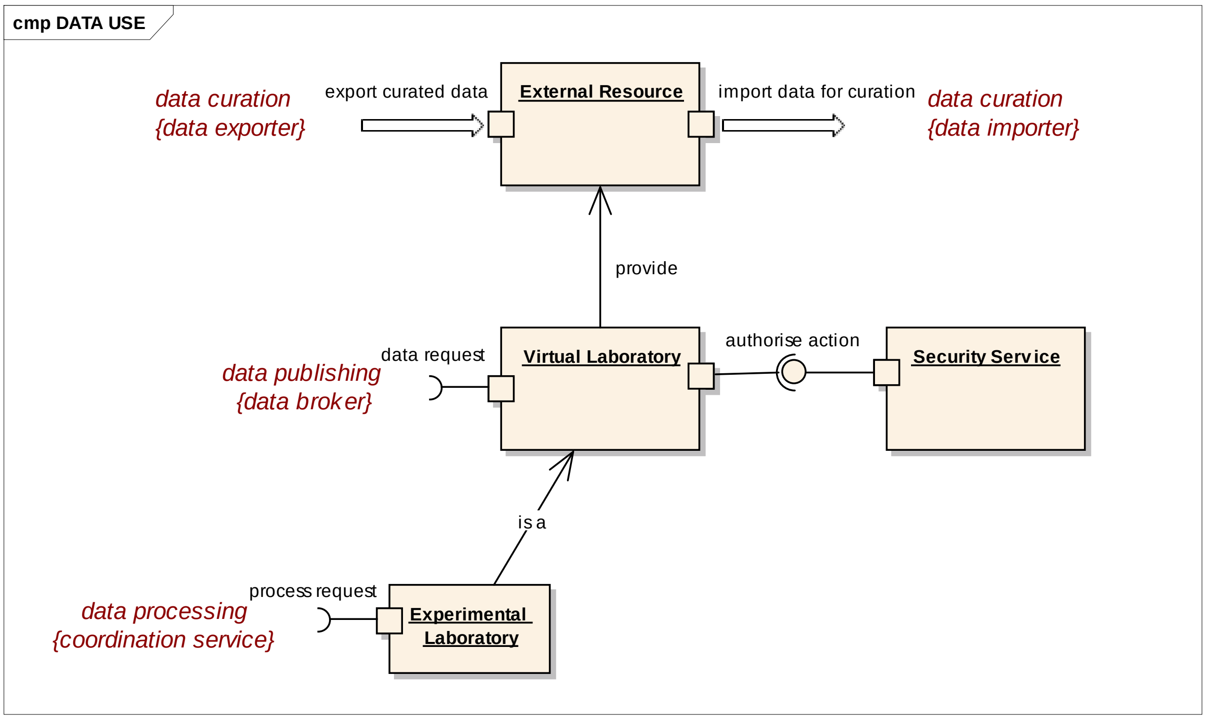


Figure 6 - Data Use

Complex interactions between the components facilitating data use and other components are mediated by **virtual laboratories** which provide a persistent context for such interactions between certain groups of users and particular components within the data management platform.
A **virtual laboratory** object encapsulates interaction between a user or group of users and a subset of the functions provided by the data management platform. Its role is to bind a security service with (potentially) any number of other platform objects.
An experimental laboratory represents a utility or a tool which allows users (researchers) to deploy datasets for processing and acquire results from computational experimentation.
All laboratories must interact with a **security service** in order to authorize requests and authenticate users of the laboratory before they can proceed with any privileged activities.

5. ARCHITECTURAL DESIGN OF THE EMSODEV DATA MANAGEMENT PLATFORM

5.1. General Description of the System

Accurate, consistent, comparable, long-term measurements of ocean parameters are key to address urgent societal and scientific challenges such as climate change, ocean ecosystem disturbance, and marine hazards. Collecting ocean data and analyzing them on a day-to-day basis are two big challenges to face in order to continuously monitor environmental processes including natural hazards, climate change, and marine ecosystems.

The EMSODEV Data Management Platform will simultaneously collect, analyze and make a number of chemical and physical ocean parameters easily accessible, addressing needs from a very large scientific user community including biologists, geoscientists, chemists, and engineers.
The platform enables the exploitation of e-infrastructure capabilities with the final goal of developing flexible and scalable data management services for long-term, high-resolution, (near)-real-time monitoring of environmental processes (e.g. natural hazards, climate change and marine ecosystems). As explained in section 4, the EMSODEV DMP includes a set of common services, compliant to the phases of the computational viewpoint of the ENVRI Reference Model v2.0 [R5] namely:

- ***data acquisition***;
- ***data curation*** (including data storage and partitioning, data quality checking and cataloguing services, import/export utilities, query services);
- ***data publishing*** (query preparation, preparation for import/export of curated data);
- ***data processing services*** (real time and/or batch processing computing capabilities);
- ***data use*** (platform authentication and authorization).

The following figure shows all the architectural components of the EMSODEV DMP included within the respective ENVRI RM phase which each component belongs to. Each EMSODEV DMP architectural component is also mapped to the computational objects described in section 4 (in *italics* for each box).

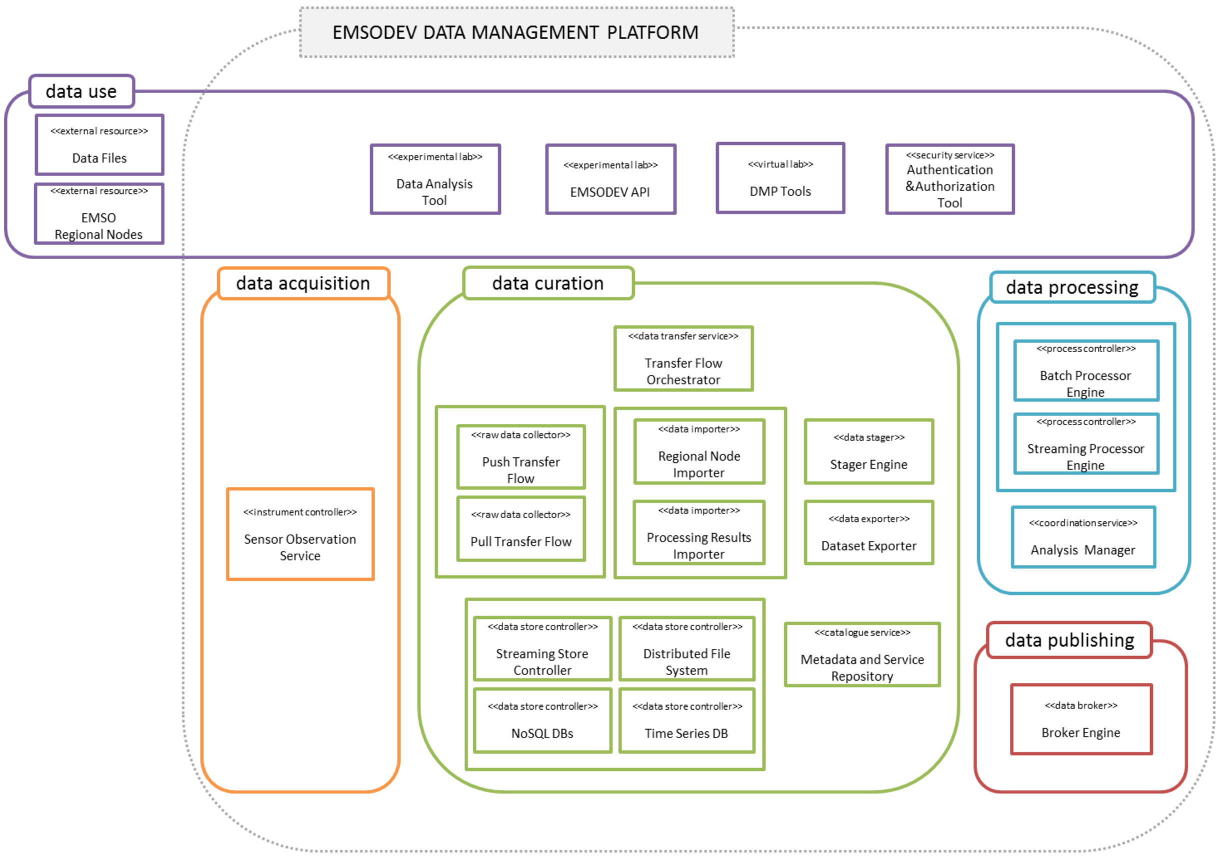

EMSODEV DATA MANAGEMENT PLATFORM
data use
<<external resource>> Data Files
<<external resource>> EMSO Regional Nodes
<<experimental lab>> Data Analysis Tool
<<experimental lab>> EMSODEV API
<<virtual lab>> DMP Tools
<<security service>> Authentication &Authorization Tool
data acquisition
<<instrument controller>> Sensor Observation Service
data curation
<<data transfer service>> Transfer Flow Orchestrator
<<raw data collector>> Push Transfer Flow
<<raw data collector>> Pull Transfer Flow
<<data importer>> Regional Node Importer
<<data importer>> Processing Results Importer
<<data stager>> Stager Engine
<<data exporter>> Dataset Exporter
<<data store controller>> Streaming Store Controller
<<data store controller>> Distributed File System
<<data store controller>> NoSQL DBs
<<data store controller>> Time Series DB
<<catalogue service>> Metadata and Service Repository
data processing
<<process controller>> Batch Processor Engine
<<process controller>> Streaming Processor Engine
<<coordination service>> Analysis Manager
data publishing
<<data broker>> Broker Engine

Figure 7 - EMSODEV DMP Architectural Components

5.2. DMP Architectural Component interactions

5.2.1. Raw Data Collection

Sensor data will be coming both in asynchronous/batch (PULL mode) and real-time mode (PUSH mode), from the **Sensor Observation Service** deployed close to each EGIM.
Sensor data can be either retrieved via APIs exposed by an SOS server (**Pull Transfer Flow** using the *retrieve data* interface) or sent to the data management platform before being consolidated on the SOS server itself (**Push Transfer Flow** which accepts data streams through the *deliver raw data* interface). Both **Push Transfer Flow** and **Pull Transfer Flow** architectural components are initiated by a **Transfer Flow Orchestrator** component.
Two main processes happen during the each transfer flow:

- data scraping, extracting parts of a body of a marine observations coming/retrieved from the SOS (Sensor Observation Service) server;
- data munging/wrangling, converting data from a "raw" format of the observations coming/retrieved from the SOS server into another one that allows data to be more conveniently consumed later on in the data chain.

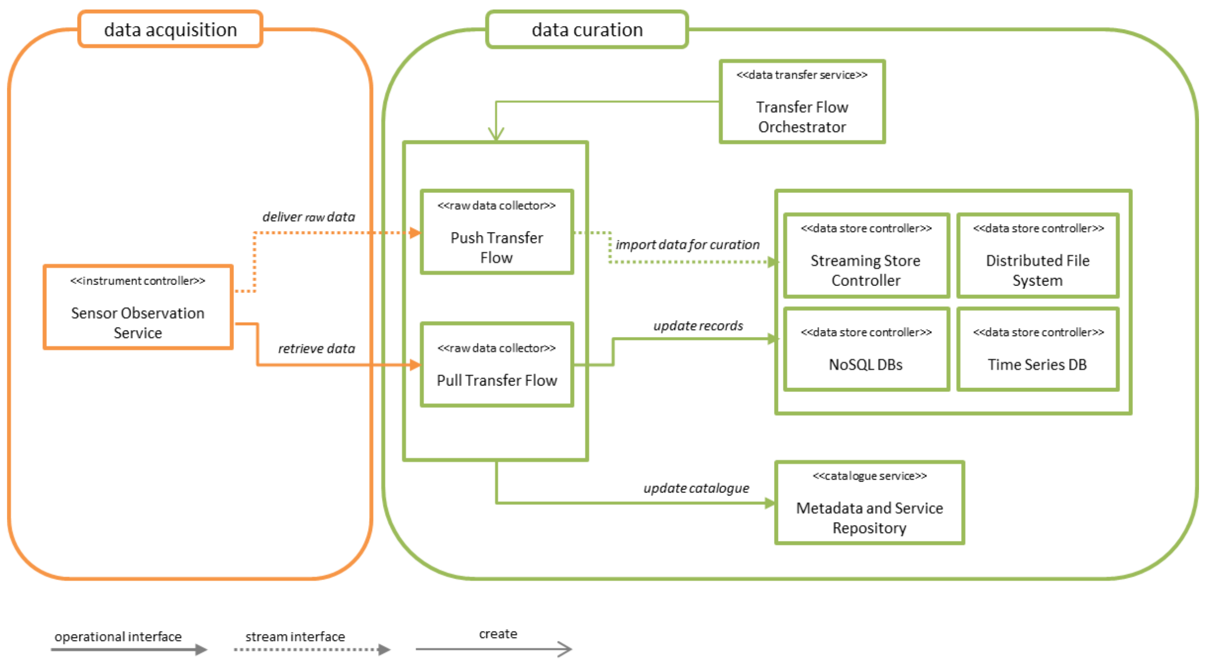


Figure 8 - Raw Data Collection component interaction

Both **Push Transfer Flow** and **Pull Transfer Flow** then send formatted data to different data store controllers (a **Distributed File System**, two different NoSQL databases (**NoSQL DBs**), a **Time Series DB** and a **Streaming Store Controller**) by HTTP POST/PUT calls against those systems (*import data for curation* stream interface for PUSH and *update records* interface for PULL). The **Distributed File System** represents the way to store (and make available) long series of historical data. The data that flow into the **Time Series DB** are stored for further analysis and real time visualization by customizable dashboards. The data that flow into NoSQL databases (**NoSQL DBs**) are used to feed the data management platform APIs which allow interested scientific communities to access various oceanographic parameters in different time ranges (e.g. the sea water temperature in a specific observatory within a certain time range).

At the same time, both **PUSH** and **PULL transfer flow** save the metadata related to a series of observations into the **Metadata and Service Repository** (by using its *update catalogue* interface).

5.2.2. User request brokering

The EMSODEV data management platform will be equipped with different tools (**DMP ToolS**) which will provide its scientific users with the following functionalities:

- activate the process of importing a dataset from external data sources such as the EMSO regional nodes;
- querying data curated within the EMSODEV data management platform;
- activate the process of defining (e.g. selecting a time range and a measured parameter) and generating a dataset to be exported outside the EMSODEV data management platform.

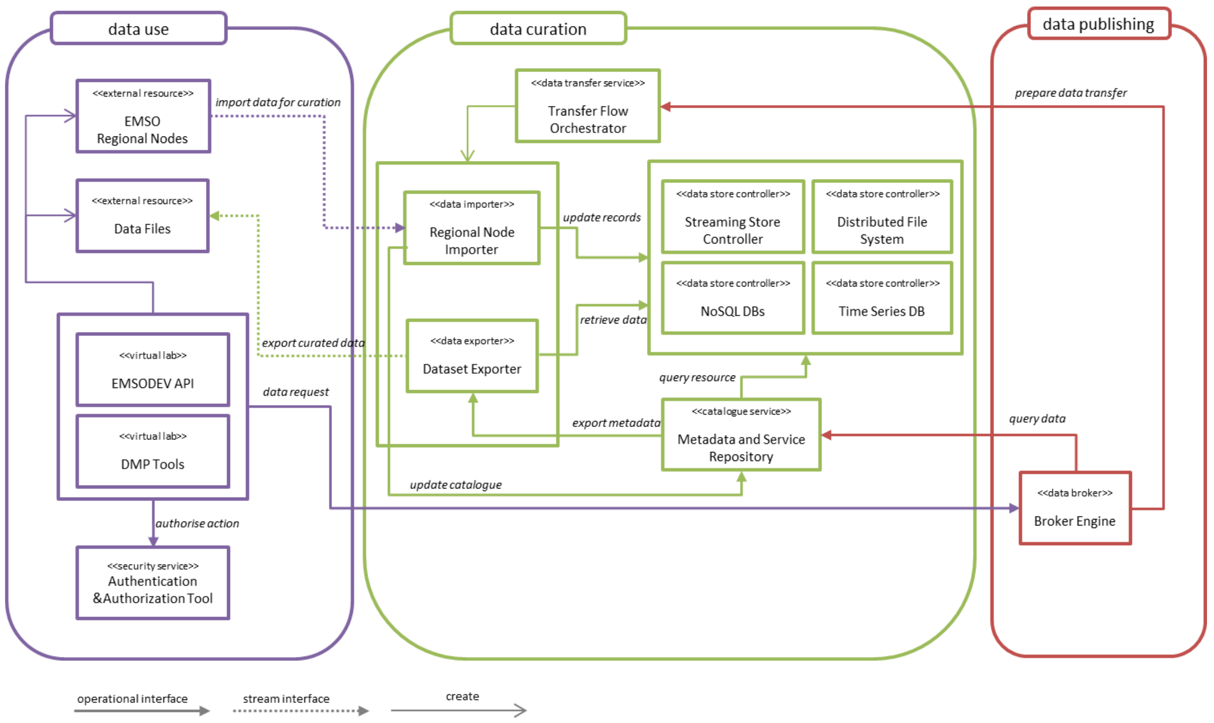


Figure 9 - User request brokering component interaction

In addition, the platform will expose its own API (**EMSODEV API**) which will allow scientific users and other European initiatives in the ocean sciences to interface with the data available within the DMP.

Regional nodes data import
The **DMP Tools** sends a request to the **Broker Engine** (via its *data request* interface) to activate the import of data from **EMSO Regional Nodes**. These external resources are located within the **DMP Tools** using the generic *create* interface (e.g. by specifying a URI and a protocol) and have been previously configured to be accessed from the DMP. At this aim, the **Broker Engine** instructs the **Transfer Flow Orchestrator** to prepare the **Regional Node Importer** which is in charge to import raw data from **EMSO Regional Nodes**; the **Regional Node Importer** will open a data-flow from an external resource (e.g. a compressed file or a local database located at an EMSO Regional Node) to one or more suitable data store controllers of the data management platform and update records within these stores as appropriate (using the *update records* interface exposed by the controllers); the **Regional Node Importer** is also responsible for registering metadata into the **Metadata and Service Repository** (by using its *update catalogue* interface).

Query curated data
The data management platform users may also use either the **DMP Tools** or the **EMSODEV API** when they need to perform a search over the curated data. In such a case, either the dashboard or the API sends a request to the **Broker Engine** (via its *data request* interface) to invoke the **Metadata and Service Repository** via its *query data* interface. It will then locate the datasets needed to answer any given query and then proceed to *query resources* within the platform *data stores*.

Export curated data
When curated data needs to be made available to external users/systems, the data management platform users take advantage of the **DMP Tools** to send a request to the **Broker Engine** (via its *data request* interface) to activate the export of data from one of the platform data store controllers. To

do so, the **Broker Engine** instructs the **Transfer Flow Orchestrator** to prepare the **Dataset Exporter** which is in charge to retrieve data from all necessary data stores (via the *retrieve data* interface), to include the integration of any additional metadata or provenance information (via the *export metadata* interface exposed by the **Metadata and Service Repository**), and finally opens a data-flow from data stores to *export the curated data* to the **Data Files** (an external resource whose location has been previously identified in the **DMP Tools** using the generic *create* interface).

5.2.3. Batch and real-time processing

The **Data Analysis Tool** originates the request (via the *process request* interface) for a data processing to be launched and managed by the **Analysis Manager**. It interprets the request and coordinates any processing workflow. In the case where data needs to be staged onto the execution platform (either the **Batch Processor Engine** or the **Streaming Processor Engine**) from a curated data store, the **Analysis Manager** will request the **Transfer Flow Orchestrator** to activate the **Stager Engine** (via the *prepare data transfer* interface).

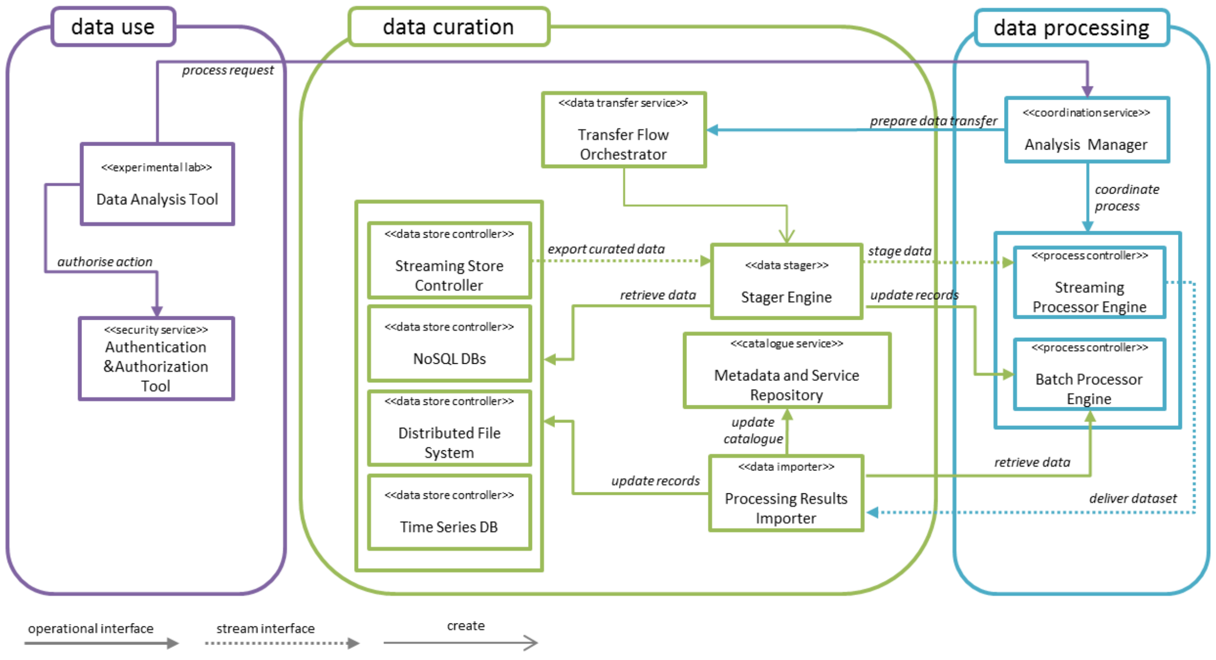

data use
data curation
data processing
process request
<<experimental lab>> Data Analysis Tool
authorise action
<<security service>> Authentication &Authorization Tool
<<data transfer service>> Transfer Flow Orchestrator
prepare data transfer
<<coordination service>> Analysis Manager
coordinate process
<<data store controller>> Streaming Store Controller
<<data store controller>> NoSQL DBs
<<data store controller>> Distributed File System
<<data store controller>> Time Series DB
export curated data
retrieve data
<<data stager>> Stager Engine
stage data
update records
<<process controller>> Streaming Processor Engine
<<process controller>> Batch Processor Engine
<<catalogue service>> Metadata and Service Repository
update catalogue
<<data importer>> Processing Results Importer
update records
retrieve data
deliver dataset
operational interface
stream interface
create

Figure 10 - Batch and real-time processing component interaction

The **Stager Engine** retrieves data to be used for batch processing from the data store controllers (with the exception of the Streaming Store Controller) via the *retrieve data* interface. It then updates records on the **Batch Processor Engine**'s side.

Data to be used for real-time processing flows from the **Streaming Store Controller** to the **Stager Engine** through the *export curated data* stream interface. It is then placed onto a staging area within the **Streaming Processor Engine** through the *stage data* stream interface.

In either modes, a data-flow is established between all required *data stores* and the processor engines. The latter represent the execution platform components on which firstly processing algorithms will be prepared (by the **Analysis Manager** via the *coordinate process* interface) and then real-time/batch data processing executed.

At the end of the real-time processing, the **Streaming Processor Engine** delivers real-time processing results to the **Processing Results Importer** (via the *deliver dataset* stream interface). In contrast, at the end of the batch processing, the **Processing Results Importer** retrieves the batch processing results using the *retrieve data* interface exposed by the **Batch Processor Engine**.

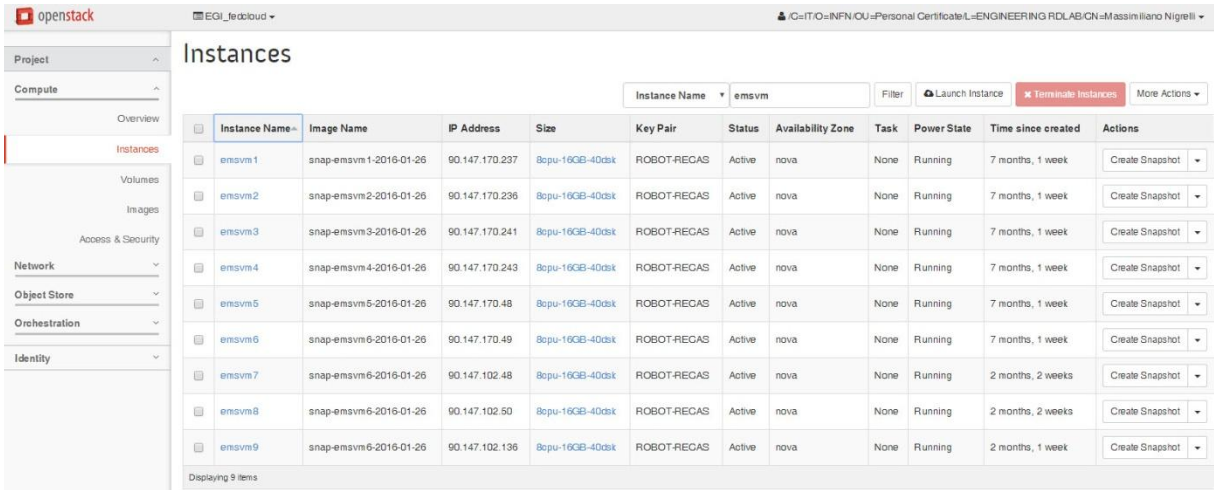
The **Processing Results Importer** finally establishes a data-flow from the execution platform underlying the processor engines to suitable *data stores* needed to host the "new" data, ensures that metadata of the newly ingested data are correctly saved (via the *update catalogue* interface exposed by the **Metadata and Service Repository**) and finally properly recorded (via the *update records* interface of data store controllers) on the data stores.

5.3.IT Infrastructure for DMP

The EMSODEV Data Management Platform will be deployed on top of the EGI Federated Cloud [R6] with the EGI support for virtualization, storage, networking and security.
The EGI Federated Cloud is a seamless grid of academic private clouds and virtualized resources, built around open standards and focusing on the requirements of the scientific community. The result is a new type of research e-infrastructure, based on the mature federated operation services that make EGI a reliable resource for science.
The EMSODEV DMP will be integrated with the EGI Federated Cloud services and will be hosted at an EGI Resource Centre.
The platform enables the exploitation of capabilities of e-Infrastructures to develop a flexible and scalable data management service for long-term, high-resolution, (near)-real-time monitoring by providing a coordinated approach for data capturing, archiving, management and delivery based on OGC standards [R7].
The requested EGI services are defined by the following properties:

- **CloudCompute**: an 'Infrastructure as a Service' cloud environment that is offered by an EGI Resource Centre which will provide at least the following on-demand compute resources (in terms of available virtual machines) where to run any kind of workload:
    - Number of Virtual CPU cores: 80 (10 VM instances with flavour 8 CPUs + 16GB RAM + 40GB HD);
    - Memory: 160 Gb (10 VM instances with flavour 8 CPUs + 16GB RAM + 40GB HD);
    - Scratch/ephemeral storage: 400 Gb (10 VM instances with flavour 8 CPUs + 16GB RAM + 40GB HD).
- **File Storage:** An EGI Resource Provider will provide 5TB of guaranteed storage capacity.

The virtualization environment is currently being tested on an OpenStack [R8] instance (provided by the Recas Bari EGI Resource Centre) that can be controlled through: OCCI [R9], OpenStackClient [R10] or a simple web user interface which is Horizon [R11]. An example screenshot of Horizon is provided in the following figure.



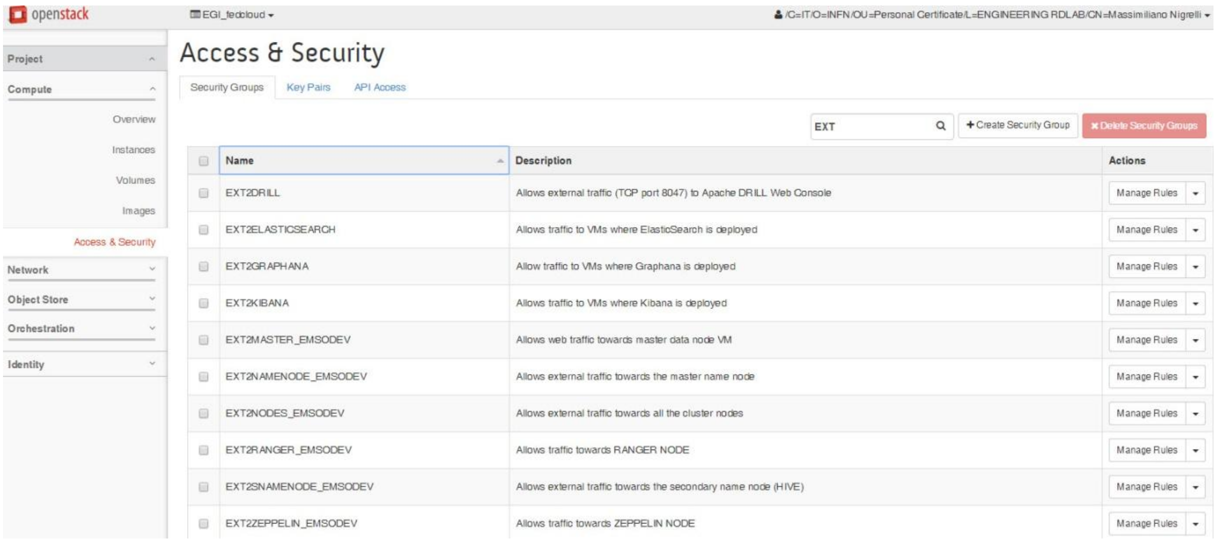
Instances

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
emsvm1	snap-emsvm1-2016-01-26	90.147.170.237	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm2	snap-emsvm2-2016-01-26	90.147.170.236	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm3	snap-emsvm3-2016-01-26	90.147.170.241	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm4	snap-emsvm4-2016-01-26	90.147.170.243	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm5	snap-emsvm5-2016-01-26	90.147.170.48	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm6	snap-emsvm6-2016-01-26	90.147.170.49	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	7 months, 1 week	Create Snapshot
emsvm7	snap-emsvm6-2016-01-26	90.147.102.48	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	2 months, 2 weeks	Create Snapshot
emsvm8	snap-emsvm6-2016-01-26	90.147.102.50	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	2 months, 2 weeks	Create Snapshot
emsvm9	snap-emsvm6-2016-01-26	90.147.102.136	8cpu-16GB-40dsk	ROBOT-RECAS	Active	nova	None	Running	2 months, 1 week	Create Snapshot

Displaying 9 items

Figure 11 - OpenStack Horizon web interface

The OpenStack virtualization platform underlying the EGI Fed Cloud also offers the chance to define the security rules which will restrict the external access to the DMP Services (e.g. only a certain service on a specific port may be accessed over the internet). Security rules (in terms of allowed IP protocol, IP and port range) are grouped into security groups (Figure 12) which may be then applied to a subset of the virtual machines where the DMP will be deployed thus allowing/inhibiting the access to its services.



Access & Security

Name	Description	Actions
EXT2DRILL	Allows external traffic (TCP port 8047) to Apache DRILL Web Console	Manage Rules
EXT2ELASTICSEARCH	Allows traffic to VMs where ElasticSearch is deployed	Manage Rules
EXT2GRAPHANA	Allow traffic to VMs where Graphana is deployed	Manage Rules
EXT2KIBANA	Allows traffic to VMs where Kibana is deployed	Manage Rules
EXT2MASTER_EMSODEV	Allows web traffic towards master data node VM	Manage Rules
EXT2NAMENODE_EMSODEV	Allows external traffic towards the master name node	Manage Rules
EXT2NODES_EMSODEV	Allows external traffic towards all the cluster nodes	Manage Rules
EXT2RANGER_EMSODEV	Allows traffic towards RANGER NODE	Manage Rules
EXT2SNAMENODE_EMSODEV	Allows external traffic towards the secondary name node (HIVE)	Manage Rules
EXT2ZEPPELIN_EMSODEV	Allows traffic towards ZEPPELIN NODE	Manage Rules

Figure 12 - OpenStack Horizon security groups definition

At the time of writing, EGI.eu (the Provider) and the EMSODEV EC project are in the process of defining a production environment for the DMP (Virtual Organization name: vo.emsodev.eu) and a related Service Level Agreement document which states needed IT resources and services to run the EMSODEV DMP over the EMSODEV project duration.

5.4. EMSODEV DMP API

This section is focuses on the specification of a Representational State Transfer Application Programming Interface (REST API) [R12][R13][R14] that allows authorised users (i.e. scientific communities) to access data managed by the EMSODEV DMP.

Fielding's dissertation [R14] gives a general explanation about the REST concepts through: "*The key abstraction of information in REST is a resource. Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a* *collection of other resources, a non-virtual object (e.g. a person), and so on. In other words, any concept that might be the target of an author's hypertext reference must fit within the definition of a resource. A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time*".

Following this principle, the EMSODEV DMP is going to be equipped with a REST API which allows an external user to access data related to EGIM nodes, instruments and parameters (the set of entities in the EMSODEV context) in a certain date range.

Resources form the nucleus of any REST API design. Resource identifiers (URI), Resource representations, API operations (using various HTTP methods), etc. are all built around the concept of Resources. A resource can be a singleton or a collection. In the EMSODEV DMP domain, (EGIM) "observatories" is a collection resource and "observatory" is a singleton resource. A resource may contain sub-collection resources. For instance in EMSODEV, the sub-collection resource "instruments" available in a certain "observatory" can be identified using the URN "`/observatories/{observatory}/instruments`".

Similarly, a singleton resource "instrument" inside the sub-collection resource "instruments" can be identified as follows: "`/observatories/{observatory}/instruments/{instrument}`".

The same approach can be followed for the collection of "parameters" measured by an {instrument} with "`/observatories/{observatory}/instruments/{instrument}/parameters`" and for the singleton resource {parameter} identified by "`/observatories/{observatory}/instruments/{instrument}/parameters/{parameter}`".

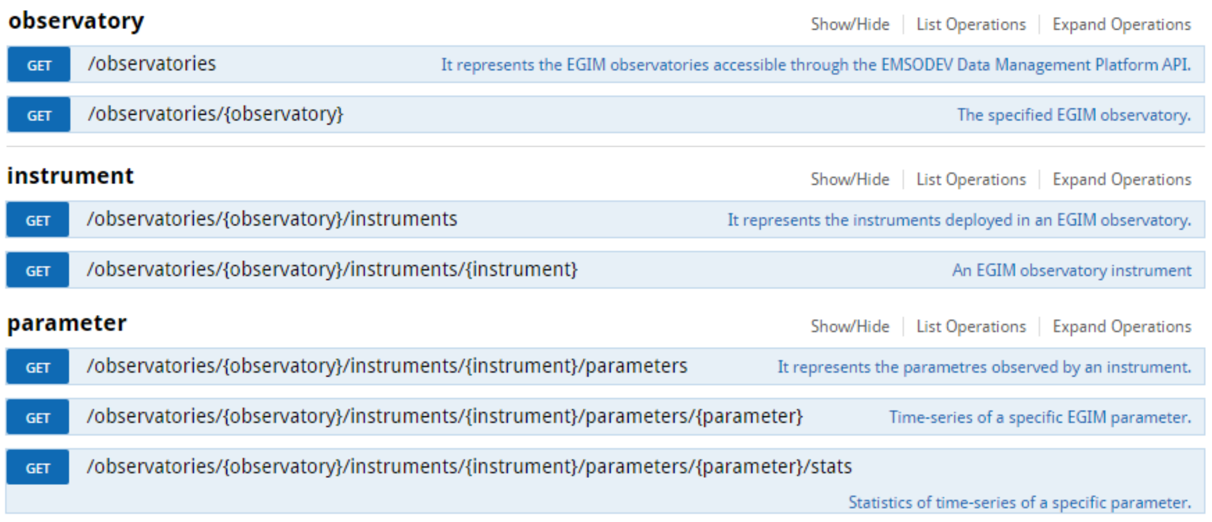
With respect to the design of the EMSODEV API, first the resources have been identified, then the interactions with the API have been modelled as HTTP verbs against these resources.

Complex result filters (e.g. restricting the time window in this case) has been designed to be implemented as query field parameters on top of the base URL.

Moreover, if we consider the "parameter" resource, the sub-resource "stats" shows possible interesting statistics of a parameter, eventually restricted over a time windows specified as query field parameters.

It is important to note that since the scope of the EMSODEV DMP API is to make data accessible to scientific communities, all the REST resources are mapped to HTTP GET verbs.

Given what was written before, the available REST API calls (under the planned https://api.emsodev.eu/ domain) will have the form shown in Figure 13.


observatory Show/Hide | List Operations | Expand Operations

GET /observatories — It represents the EGIM observatories accessible through the EMSODEV Data Management Platform API.

GET /observatories/{observatory} — The specified EGIM observatory.

instrument Show/Hide | List Operations | Expand Operations

GET /observatories/{observatory}/instruments — It represents the instruments deployed in an EGIM observatory.

GET /observatories/{observatory}/instruments/{instrument} — An EGIM observatory instrument

parameter Show/Hide | List Operations | Expand Operations

GET /observatories/{observatory}/instruments/{instrument}/parameters — It represents the parametres observed by an instrument.

GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter} — Time-series of a specific EGIM parameter.

GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter}/stats — Statistics of time-series of a specific parameter.

Figure 13 - EMSODEV Data Management Platform API

Annex A contains an exploded view of the definition of each of the API calls which includes a preliminary version of the i) implementation notes, ii) query parameters and iii) response data model. Further improvements to the data model will be defined during the development lifecycle of the API.

5.4.1. API supporting tools

Swagger [R15] and its related tools, which are compliant to the Open API Initiative (OAI) [R16] and the Open API Specification, are going to be to be used for the EMSODEV API design and development lifecycle. In particular, the following tools are going to be used for this purpose:

- **Swagger Editor** [R17]: allows for API specifications to be edited in YAML format through the web browser and documentations to be previewed in real time. Valid Swagger JSON descriptions can then be generated and used with the full Swagger tool suite;
- **Swagger UI** [R18]: a dependency-free collection of HTML, Javascript, and CSS assets that dynamically generate beautiful documentation and sandbox from a Swagger-compliant API (i.d Swagger Editor); this environment is also able to “try out” the usage of each API. The EMSODEV DMP documentation will be available through this tool under the emsodev.eu domain;
- **Swagger CodeGen** [R19]: allows for an easy and automated generation of API client libraries, server stubs starting from an API definition compliant to the OpenAPI Specification [R30], like the one produced by Swagger Editor. Several programming languages (and related framework) are supported (e.g Java, Python, Scala, etc.).

5.4.2. EMSODEV REST API Security

A REST API can be accessed through an Internet connection by using different devices with different technologies. Nevertheless not all the Internet connections and devices are secure and it may expose the API to malicious attacks; for these reasons, when an API exposes resources that contain information that can be accessed by a restricted audience, the choice of a valid API security method is an important and critical decision. In general, a RESTful API should be stateless, this means that request authentication should not depend on cookies or sessions, instead, each request should come with some sort of authentication credentials. Developing from scratch a security system can be very expensive in terms of development and maintenance. Moreover, end-users do not want to share their credentials with the resources server that hosts the data. For these reasons, a consolidated and easy design solution to go beyond this limit is needed.

A token-based authentication/authorization mechanism helps separating the role of the client from the role of the resource owner by introducing an authentication/authorization layer.

The following image shows the flow of a typical token-based authentication/authorization mechanism.

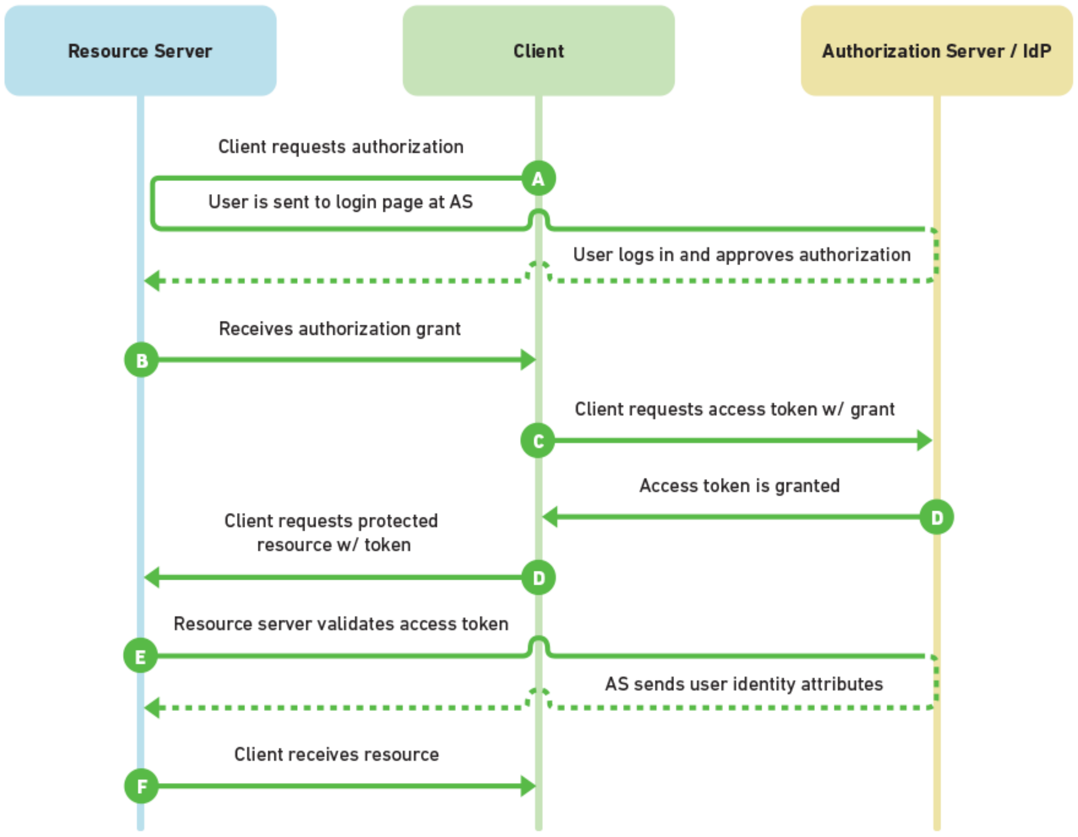

Resource Server
Client
Authorization Server / IdP
Client requests authorization
A
User is sent to login page at AS
User logs in and approves authorization
Receives authorization grant
B
Client requests access token w/ grant
C
Access token is granted
D
Client requests protected resource w/ token
D
Resource server validates access token
E
AS sends user identity attributes
Client receives resource
F

Figure 14 - Token-based authentication/authorization mechanism

6. DATA MANAGEMENT PLATFORM TOOLS

6.1. Module for Ocean Observatory Data Analysis (MOODA)

The Module for Ocean Observatory Data Analysis (MOODA) is a software with a Graphical User Interface (GUI) developed for scientists. The software helps to facilitate Data Access (mainly off-line) for further analysis by the scientific community.

Some of the features the MOODA offers are:

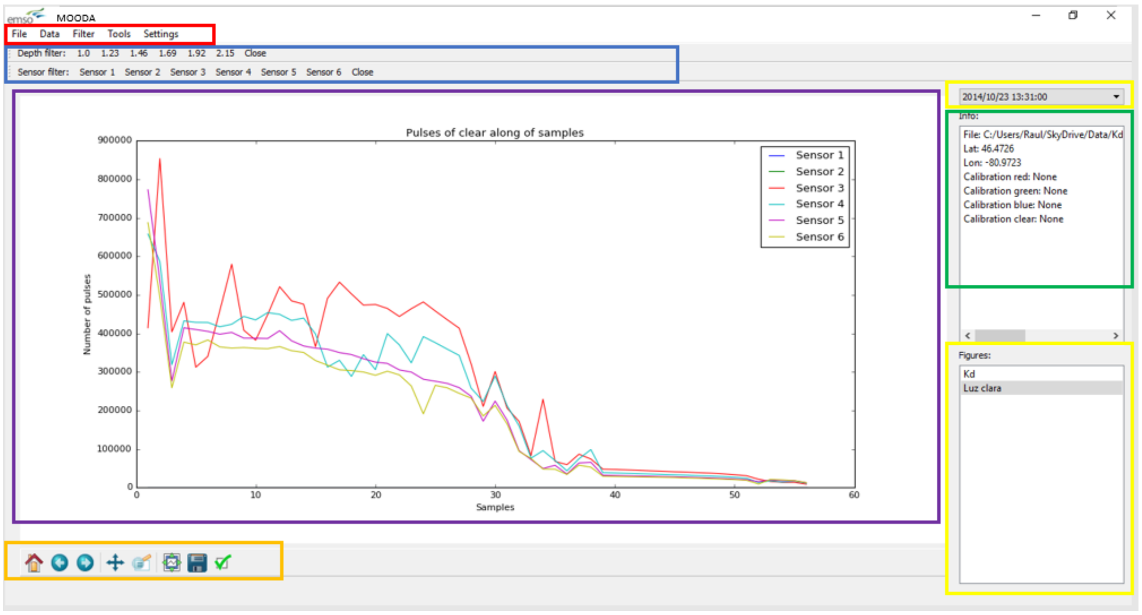
- Direct data access with complex query capabilities;
- Data filtering methods based on metadata information;
- Complex visualization tools;
- Summary reports of the validated data generated from a specific query, including event annotations. These reports (with graphical representations) will be used for a first global evaluation of the data available for a further scientific analysis;
- Specific data analysis tools for different scientific disciplines;
- The system will be designed to be open, adaptable and scalable allowing future contributions from researchers and developers from all the disciplines associated to the EMSO observatories.

MOODA aims to make informative plots as a central part of exploring and understanding data.

We are developing MOODA in Python, a language with fully open source tools for development that is gaining popularity in the scientific community.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages for a wide range of applications. There a modules for *visualization* (such as *matplotlib* [R20]), *scientific computing* (SciPy [R21] or *Pandas* [R22]), *data structures* and *statistical routines* from SciPy and statsmodels which encourages program modularity and code reuse [R23].

For example, MOODA visualization capabilities have been conceived using mainly the matplotlib module, and the graphics can be further tweaked using matplotlib tools and rendered with any of the matplotlib backends to generate publication-quality figures. We have already developed a very preliminary version of the software. Although the layout structure could change during the development, the software will contain the same parts as the current version. Figure 15 shows the first version of the MOODA (work in progress).


MOODA
File Data Filter Tools Settings
Depth filter: 1.0 1.23 1.46 1.69 1.92 2.15 Close
Sensor filter: Sensor 1 Sensor 2 Sensor 3 Sensor 4 Sensor 5 Sensor 6 Close
Pulses of clear along of samples
Number of pulses
Samples
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Sensor 5
Sensor 6
2014/10/23 13:31:00
Info:
Figures:
Kd
Luz clara

File: C:/Users/Raul/SkyDrive/Data/Kd
Lat: 46.4726
Lon: -80.9723
Calibration red: None
Calibration green: None
Calibration blue: None
Calibration clear: None

Figure 15 - MOODA prototype v0.1. Although there are a number of ways to construct this layout, we will define a model that includes seven major elements: 1) the main window holds everything together, 2) a list of figures and data sets (yellow), 3) a plot container (purple), 4) a data information box (green), 5) a navigation menu (red), 6) some data filter toolbars (blue) and 7) a plot toolbar (orange).

The following sub-sections describe the key features of MOODA.

6.1.1. Direct data access with complex query capabilities

In the simplest terms, an application program interface (API) is a sets of requirements that govern how one application can talk to another. Generally, APIs are not human-friendly (at least, not for people without an IT knowledge).

MOODA will use the API from the EMSO Data Base to receive specific data. The software will help users to retrieve data with a set of intuitive window options, minimizing the need for understanding the internal management of the API.

MOODA could also use the data previously downloaded to the user’s computer.

6.1.2. Data filtering methods based on metadata information

To analyse the enormous amount of data provided from the EMSO Data Base, users will need data filters, prioritizing data and making efficient analysis to alleviate the problem of information overload.

MOODA will help to filter data through window options and buttons.

6.1.3. Summary reports

MOODA will have the option to generate documentation with the figures and the most relevant information of the analysed data. This option will be optional, but it will help scientists to save the figures and the relevant information from a set of data before starting to analyse another one.

6.1.4. Expandable data analysis and visualization tools for different scientific disciplines

Python supports multiple options for visualizing data. Because of this variety, it can be challenging to figure out which one to use when.

MOODA will provide a set of example graphs to understand data. Users could select or modify the graphs with a visual interface.

6.1.5. Open-source software

MOODA will be an open-source software written in Python.

We are developing software following the PEP8 directives [R25]. PEP8 is a document that explains the conventions for the Python code comprising the standard library in the main Python distribution. This type of style of writing code will help other people to understand the code and make important contributions.

The source code of the MOODA and the complete application could be download from the EMSO web page and GitHUB [R26].

6.2.Sensor Monitoring Dashboard (SMD)

The Sensor Monitoring DashBoard (SMD) is a web tool for accessing, maintenance and operation of all sensors recording data on each EMSO node connected to the entire system.

The entry point to this tool will be a web Interface with restricted access for visualisation or/and administrative tasks.

The SMD Web Interface will be able to add EMSO node LabMonitor (Zabbix) hosts in a friendly graphical manner, only by entering URL to EMSO node LabMonitor, user and password. In this way, once the LabMonitor host is configured, we can monitor and/or configure all the parameters referring to acquisition of sensor data, alarms, and triggers.

All the data that must be registered as an event or alarm to be accessed for the Data Management Layer should be sent to the corresponding SOS Server using the SOS interface at each EMSO node.

In figure 14, all components related with SMD core system are graphically detailed.

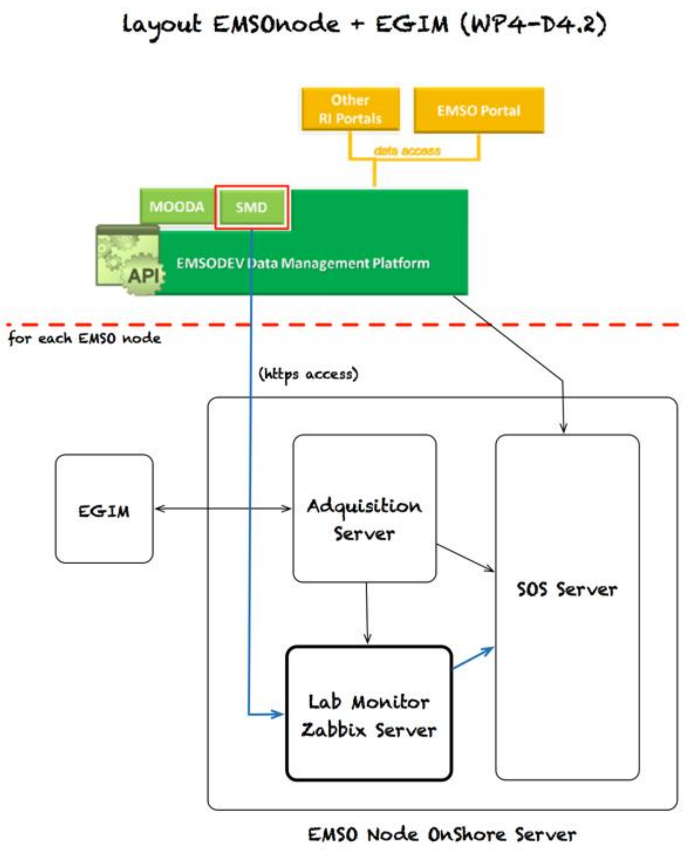


Figure 16 - The SMD core system.

7. INTEROPERABILITY WITH EXISTING EMSO REGIONAL DATA NODES

Due to the distributed structure of EMSO, which builds on national and regional infrastructures, the data management of EMSO'S regional data nodes is not centrally organized. Instead, already available data management infrastructures and archives are used. These are operated autonomously by each regional node. This concept and the necessary technological and organisational prerequisites have successfully been established within the ESONET NoE and shall now be operationally continued within EMSODEV.

Trustworthy data centres such as World Data Centres (WDC) and other certified or otherwise mandated data centres like National Data Centres (IOC/IODE) are the backbone for EMSO's data archiving strategy and are responsible for the long term archiving of data collected by each regional node's observatories and ensure long term availability of EMSO data. The integration of these archives requires considerable efforts regarding standardisation and synchronisation of data, metadata and workflows. Consequently, the main challenge for the integration of EMSO's regional data nodes is to agree on an interoperability framework based on international standards necessary to implement the agreed data management policies and work flows within a distributed data infrastructure. Therefore, EMSO strongly cooperates with other initiatives and integrators such as ATLANTOS, FixO3, JERICHO+, ENVRI+, SeaDataNet as well as EMODnet. Some common standards already are generally accepted by these communities.

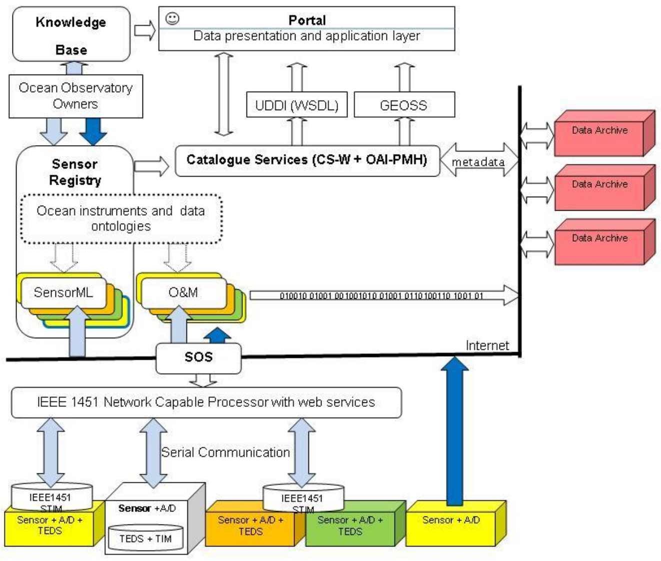


Figure 17 - Overview of the main standard components used for the ESONET technical infrastructure.

A generic technological architecture of such a distributed system is already defined within the ESONET NoE (Figure 17).

7.1. Access to Archived Data

Due to the distributed nature of EMSO, integration of this data will primarily be enabled via the integration of metadata while the corresponding data will remain under control of each node's data archive. Within the marine community several approaches currently exist to provide access to metadata as well as data. The most basic approach is to offer data files with integrated metadata in NetCDF format via FTP servers. This approach has been chosen by OceanSites as well as EuroSites and currently is used by EuroGOOS. These intitiatives have collected a significant amount of relevant EMSO historical oceanographic data. Metadata extraction methods already exist and can be integrated within EMSODEV. Beyond this simple approach, systems such as INGV's MOIST or PANGAEA offer access to metadata via OAI-PMH or OGC-CSW and provide metadata descriptions according to common standards such as the INSPIRE compliant ISO19139, Global Change Master Directory - Directory Interchange Format (GCMD DIF) format or Dublin Core format.

All EMSO relevant metadata will be harvested at regular intervals using panFMP (PANGAEA Framework for Metadata Portals) [R31] as the technical indexing and cataloguing platform. Several user frontends and search interfaces for panFMP already exist. Some of these have already been used for ESONET and the previous EMSO data portal. Technically, this is achieved by automatically harvesting the provided metadata files and creating an index, which is the base for the full text search offered to the end user. When a user enters a search term and/or other constraints (i.e., temporal or geographical limits), the index is being searched accordingly. Matching results will be presented as a combination of a URL and an accompanying, summarizing description.

7.2.Access to Real Time Data

EMSO will follow core standards of the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) suite of standards, namely the OGC standards SensorML, Catalogue Service for Web (CSW), Sensor Observation Service (SOS) and Observations and Measurements (O&M). As the above mentioned SWE standards represent a generic and abstract framework other than detailed implementation rules, they allow much interpretative freedom. Therefore, to ensure community wide compatibility, EMSO has joined the 'marine SWE group'21 to define or decide on standard ontologies and to elaborate application profiles.

SensorML is an eXtensible Markup Language (XML) for describing sensor systems and processes that can be used to feed a Sensor Registry, the common catalogue of ocean observatory sensors which could be accessed via an OGC CSW interface. Access to the EMSO regional nodes real-time observatory data will be provided by means of SOS servers which have been implemented for cabled observatories such as ANTARES, OBSEA or the Koljoefjord demo site. Several additional SOS implementations are currently in preparation within the FixO3 project.

7.3.Interoperability with International Initiatives

To enable effective interoperability among EMSO data archives as well as with international initiatives, it is most essential to make available as much information about technical capabilities of EMSO as possible. All available services shall therefore be registered at the Global Earth Observation System of Systems (GEOSS), which operates the GEOSS Components and Services Registry (CSR). These entries are available to the scientific community as ISO 19115 format and can be reused accordingly. To ensure interoperability with EMODnet and other initiatives, the same formats and standards which are suitable for internal data and metadata exchange are appropriate for EMODnet integration. Formal negotiations with EMODnet are needed to initiate the integration process which will make use of the EMSODEV infrastructure as well as the distributed EMSO regional node infrastructure as described above.

[21] See: Jirka et al. (2016) Marine Profiles for OGC Sensor Web Enablement Standards: http://meetingorganizer.copernicus.org/EGU2016/EGU2016-14690.pdf

8. FINAL CONSIDERATIONS

The EMSODEV Data Management Platform Architecture relies on some of the most technologically advanced tools and solutions available to market and scientific communities. By the architecture of services described in this report the whole EMSODEV community fosters the adoption of the most powerful innovation technologies by Ocean Sciences.

The aim of the work done was twofold:

- Going further ahead on the strong technological improvement path already followed by other projects involving Ocean Science actors in Europe
- Demonstrate how ICT-based innovation can improve fruition of EGIM and Data produced by EMSODEV and regional nodes in an easy-to-use manner.

The outcomes of T6.2 will be the basis for the Data Management Platform development task (T6.3). This will provide a pilot implementation and opportunity to provide in-the-field demonstration of capabilities and performance evaluation.

9. REFERENCES

[R1] EMSODEV Project Description of Action.

[R2] EMSODEV Project Deliverable D2.1. "REPORT ON WORKSHOP ON SCIENTIFIC REQUIREMENTS"

[R3] EMSODEV Project Deliverable D6.1. "Data Management Plan"

[R4] "Technology readiness levels (TRL)" (PDF). European Commission, G. Technology readiness levels (TRL), HORIZON 2020 – WORK PROGRAMME 2014-2015 General Annexes, Extract from Part 19 - Commission Decision C(2014)4995.

[R5] ENVRI Reference Model Overview v2.0 - https://confluence.egi.eu/display/EC/ENVRI+Reference+Model

[R6] EGI Federated Cloud - https://www.egi.eu/infrastructure/cloud/ [Accessed 09/08/2016]

[R7] Open Geospatial Consortium - http://www.opengeospatial.org/standards [Accessed 09/08/2016]

[R8] OpenStack - https://www.openstack.org/ [Accessed 09/08/2016]

[R9] OCCI Open Cloud Computing Interface - http://occi-wg.org/ [Accessed 09/08/2016]

[R10] OpenStackClient - http://docs.openstack.org/developer/python-openstackclient/ [Accessed 09/08/2016]

[R11] Horizon: The OpenStack Dashboard Project - http://docs.openstack.org/developer/horizon/ [Accessed 09/08/2016]

[R12] What is REST API - http://restfulapi.net/ [Accessed 09/08/2016]

[R13] REST - https://en.wikipedia.org/wiki/Representational_state_transfer [Accessed 09/08/2016]

[R14] Roy Thomas Fielding. 2000. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation. University of California, Irvine. AAI9980887.

[R15] Swagger – http://swagger.io/ [Accessed 09/08/2016]

[R16] Open API Initiative - https://openapis.org/ [Accessed 09/08/2016]

[R17] Swagger Editor - http://swagger.io/swagger-editor/ [Accessed 09/08/2016]

[R18] Swagger UI - http://swagger.io/swagger-ui/ [Accessed 09/08/2016]

[R19] Swagger CodeGen - http://swagger.io/swagger-codegen/ [Accessed 09/08/2016]

[R20] Matplotlib - http://matplotlib.org/ [Accessed 09/08/2016]

[R21] SciPy - https://www.scipy.org/ [Accessed 09/08/2016]

[R22] Pandas - http://pandas.pydata.org/ [Accessed 09/08/2016]

[R23] Python - https://www.python.org/ [Accessed 09/08/2016]

[R24] St. Laurent, Andrew M. (2008). Understanding Open Source and Free Software Licensing. O'Reilly Media. p. 4. ISBN 9780596553951.

[R25] PEP8 Style Guide for Python Code, https://www.python.org/dev/peps/pep-0008/ [Accessed 09/08/2016]

[R26] GitHUB - https://github.com [Accessed 09/08/2016]

[R27] Mario Bunge. 2003. Philosophical Dictionary. Amherst NY, Prometheus Books.

[R28] Frederich Hayek. 1976. The New Deal and The International Monetary System. In Watershed of Empire: Essays on New Deal Foreign Policy. Colorado Springs, Ralph Myles Publisher.

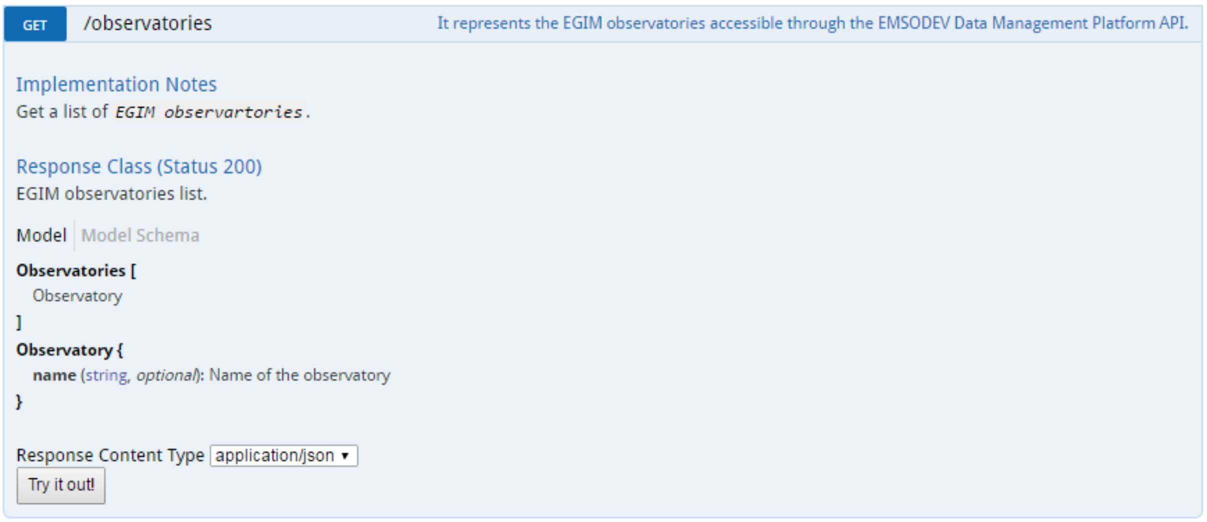
[R29] Frederich Hayek. 1933. The Trend of Economic Thinking. Economica 13.

[R30] OpenAPI Specification - https://github.com/OAI/OpenAPI-Specification [Accessed 09/08/2016]

[R31] PANGAEA Framework for Metadata Portals - http://www.panfmp.org/ [Accessed 09/08/2016]

ANNEX A. EMSODEV DATA MANAGEMENT PLATFORM API

This Annex contains an exploded view of the definition of each of the API call which includes a preliminary version of the i) implementation notes, ii) query paramenters and iii) responses data model. Further improvements to the data model will be defined during the development lifecycle of the API.


GET /observatories — It represents the EGIM observatories accessible through the EMSODEV Data Management Platform API.

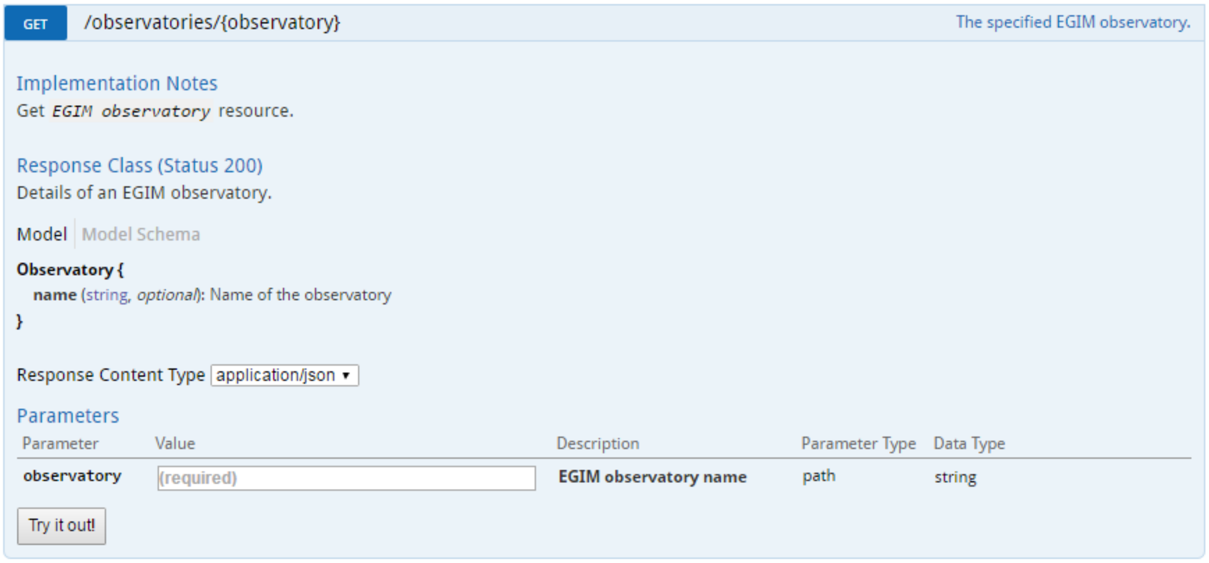
Implementation Notes
Get a list of `EGIM observartories`.

Response Class (Status 200)
EGIM observatories list.

```
Observatories [
  Observatory
]
Observatory {
  name (string, optional): Name of the observatory
}
```

Response Content Type: application/json

Figure 18 – GET /observatories


GET /observatories/{observatory} — The specified EGIM observatory.

Implementation Notes
Get `EGIM observatory` resource.

Response Class (Status 200)
Details of an EGIM observatory.

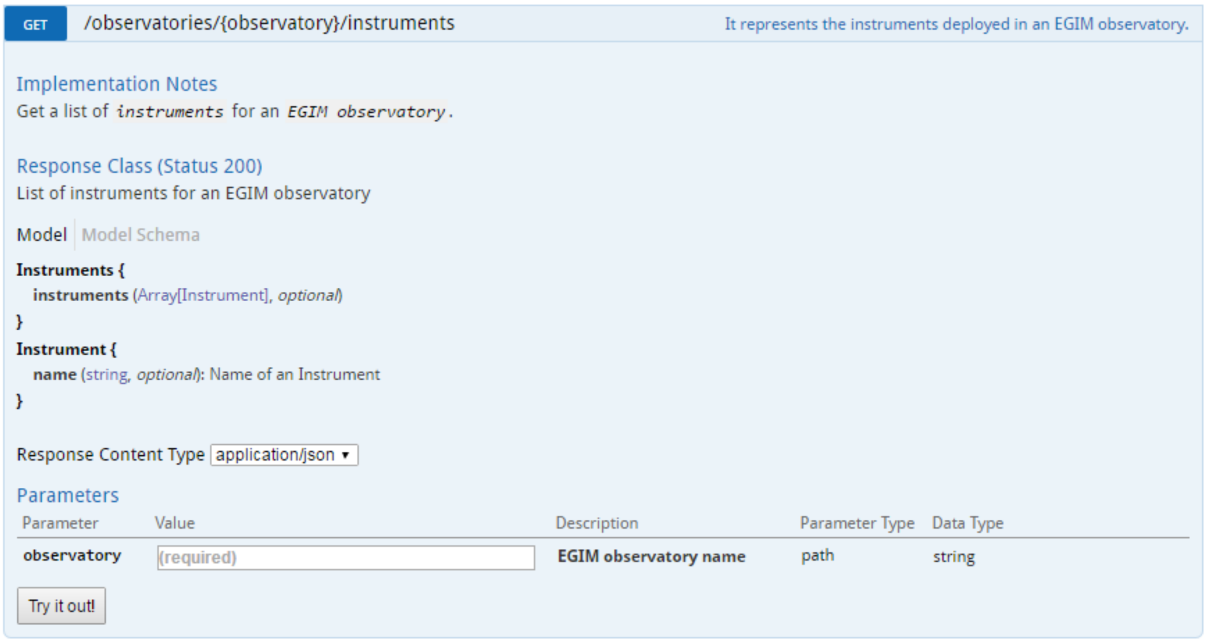
```
Observatory {
  name (string, optional): Name of the observatory
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	**EGIM observatory name**	path	string

Figure 19 - GET /observatories/{observatory}



GET /observatories/{observatory}/instruments — It represents the instruments deployed in an EGIM observatory.

Implementation Notes

Get a list of `instruments` for an `EGIM observatory`.

Response Class (Status 200)

List of instruments for an EGIM observatory

Model | Model Schema

```
Instruments {
  instruments (Array[Instrument], optional)
}
Instrument {
  name (string, optional): Name of an Instrument
}
```

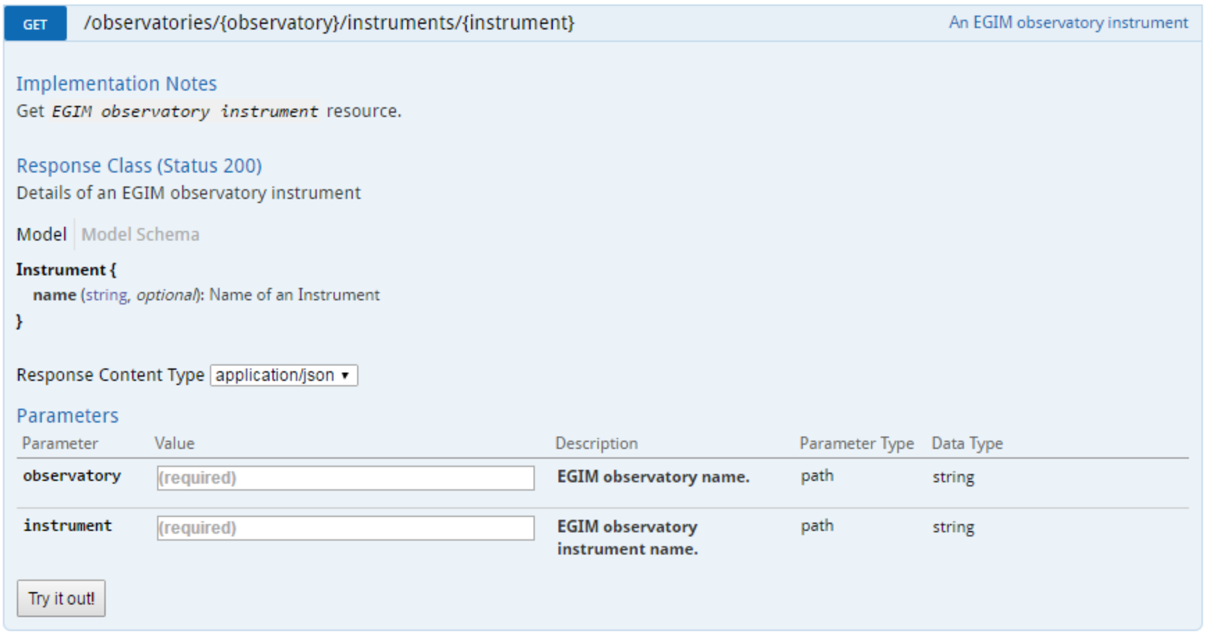
Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	**EGIM observatory name**	path	string

Try it out!

Figure 20 - GET /observatories/{observatory}/instruments



GET /observatories/{observatory}/instruments/{instrument} — An EGIM observatory instrument

Implementation Notes

Get `EGIM observatory instrument` resource.

Response Class (Status 200)

Details of an EGIM observatory instrument

Model | Model Schema

```
Instrument {
  name (string, optional): Name of an Instrument
}
```

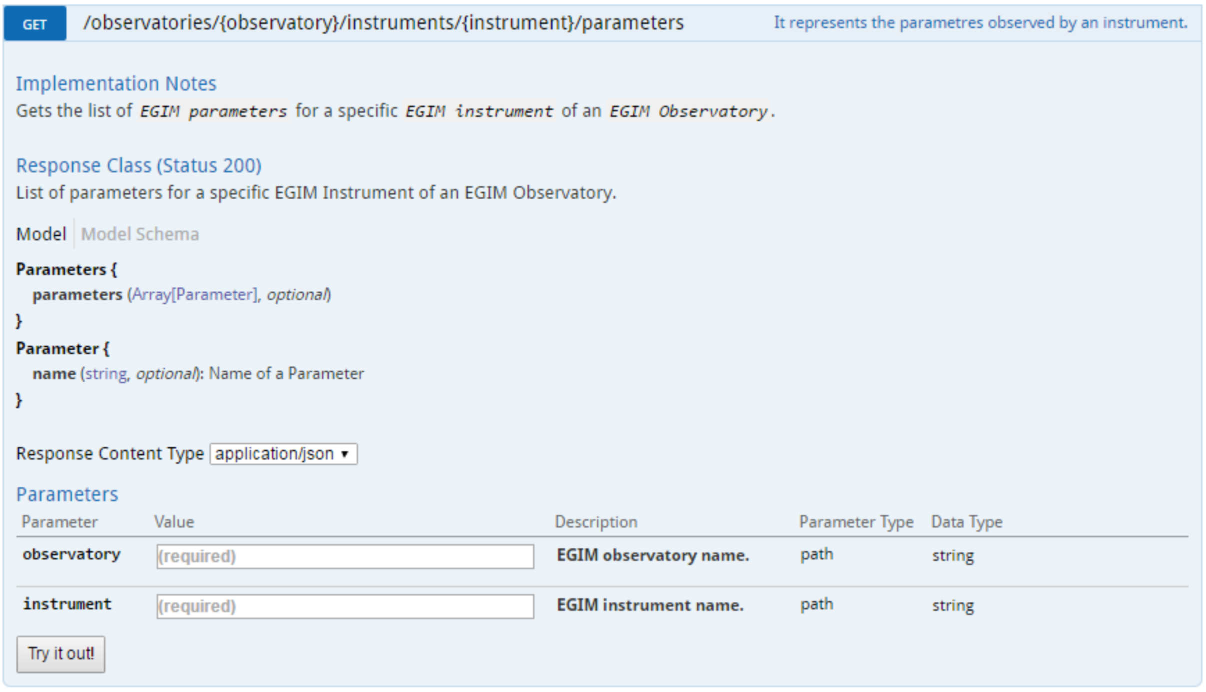
Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	**EGIM observatory name.**	path	string
instrument	(required)	**EGIM observatory instrument name.**	path	string

Try it out!

Figure 21 - GET /observatories/{observatory}/instruments/{instrument}



GET /observatories/{observatory}/instruments/{instrument}/parameters

It represents the parametres observed by an instrument.

Implementation Notes

Gets the list of `EGIM parameters` for a specific `EGIM instrument` of an `EGIM Observatory`.

Response Class (Status 200)

List of parameters for a specific EGIM Instrument of an EGIM Observatory.

Model | Model Schema

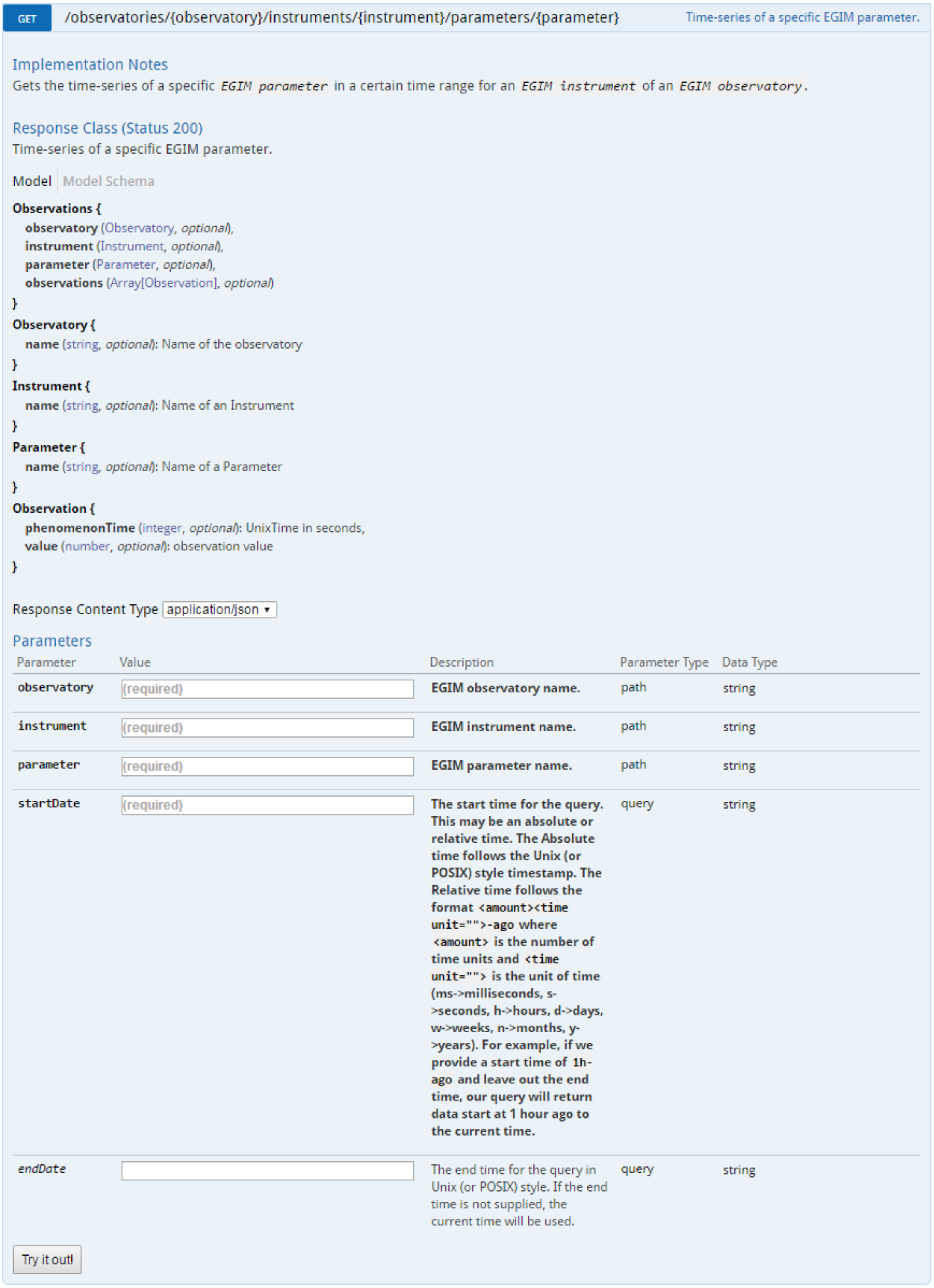
```
Parameters {
  parameters (Array[Parameter], optional)
}
Parameter {
  name (string, optional): Name of a Parameter
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	EGIM observatory name.	path	string
instrument	(required)	EGIM instrument name.	path	string

Figure 22 - GET /observatories/{observatory}/instruments/{instrument}



GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter} — Time-series of a specific EGIM parameter.

Implementation Notes

Gets the time-series of a specific *EGIM parameter* in a certain time range for an *EGIM instrument* of an *EGIM observatory*.

Response Class (Status 200)

Time-series of a specific EGIM parameter.

Model | Model Schema

```
Observations {
  observatory (Observatory, optional),
  instrument (Instrument, optional),
  parameter (Parameter, optional),
  observations (Array[Observation], optional)
}
Observatory {
  name (string, optional): Name of the observatory
}
Instrument {
  name (string, optional): Name of an Instrument
}
Parameter {
  name (string, optional): Name of a Parameter
}
Observation {
  phenomenonTime (integer, optional): UnixTime in seconds,
  value (number, optional): observation value
}
```

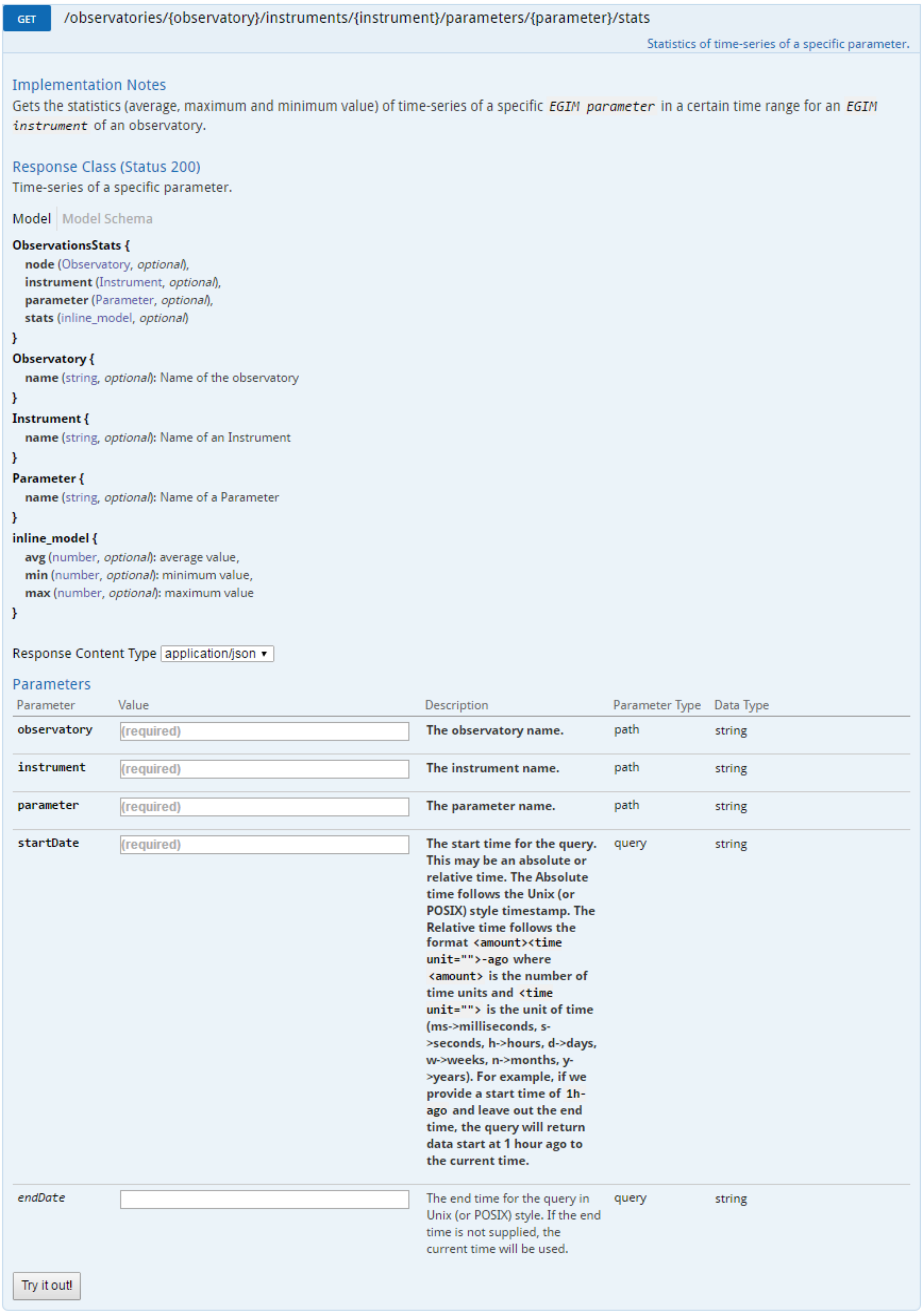
Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	EGIM observatory name.	path	string
instrument	(required)	EGIM instrument name.	path	string
parameter	(required)	EGIM parameter name.	path	string
startDate	(required)	The start time for the query. This may be an absolute or relative time. The Absolute time follows the Unix (or POSIX) style timestamp. The Relative time follows the format `<amount><time unit="">-ago` where `<amount>` is the number of time units and `<time unit="">` is the unit of time (ms->milliseconds, s->seconds, h->hours, d->days, w->weeks, n->months, y->years). For example, if we provide a start time of `1h-ago` and leave out the end time, our query will return data start at 1 hour ago to the current time.	query	string
endDate		The end time for the query in Unix (or POSIX) style. If the end time is not supplied, the current time will be used.	query	string

Try it out!

Figure 23 - GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter}



GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter}/stats

Statistics of time-series of a specific parameter.

Implementation Notes

Gets the statistics (average, maximum and minimum value) of time-series of a specific `EGIM parameter` in a certain time range for an `EGIM instrument` of an observatory.

Response Class (Status 200)

Time-series of a specific parameter.

Model | Model Schema

```
ObservationsStats {
  node (Observatory, optional),
  instrument (Instrument, optional),
  parameter (Parameter, optional),
  stats (inline_model, optional)
}
Observatory {
  name (string, optional): Name of the observatory
}
Instrument {
  name (string, optional): Name of an Instrument
}
Parameter {
  name (string, optional): Name of a Parameter
}
inline_model {
  avg (number, optional): average value,
  min (number, optional): minimum value,
  max (number, optional): maximum value
}
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
observatory	(required)	**The observatory name.**	path	string
instrument	(required)	**The instrument name.**	path	string
parameter	(required)	**The parameter name.**	path	string
startDate	(required)	**The start time for the query. This may be an absolute or relative time. The Absolute time follows the Unix (or POSIX) style timestamp. The Relative time follows the format `<amount><time unit="">-ago` where `<amount>` is the number of time units and `<time unit="">` is the unit of time (ms->milliseconds, s->seconds, h->hours, d->days, w->weeks, n->months, y->years). For example, if we provide a start time of `1h-ago` and leave out the end time, the query will return data start at 1 hour ago to the current time.**	query	string
endDate		The end time for the query in Unix (or POSIX) style. If the end time is not supplied, the current time will be used.	query	string

Figure 24 - GET /observatories/{observatory}/instruments/{instrument}/parameters/{parameter}/stats