



D9.3

RIs technical specification

Work Package	WP9
Lead partner	IFREMER
Status	Reviewed
Deliverable type	Report
Dissemination level	Public
Due date	31-03-2020
Submission date	19-05-2020

Deliverable abstract

The overarching goal of ENVRI-FAIR is that all participating research infrastructures (RIs) will improve their FAIRness and become ready for connection of their data repositories and services to the European Open Science Cloud (EOSC). Deliverable 9.1 has reported on the roadmap of the RIs in the marine subdomain towards improving their FAIRness. It presented the approach of using FAIR questionnaires (together with WP5) to identify the strengths and weaknesses of each RI and a first indicative set of activities to improve identified weaknesses or gaps. After formulation in Deliverable D9.2 of implementation plans for mitigating these gaps during the next phase of the ENVRI-FAIR project, the RIs in the marine subdomain have specified the technical implementation in this Deliverable D9.3



DELIVERY SLIP

	Name	Partner Organization	Date
Main Authors	Peter Thijsse Katrina Exter Alex Vermeulen Benjamin Pfeil Thierry Carval Ivan Rodero	MARIS LifeWatch-ERIC ICOS ERIC UIB IFREMER EMSO ERIC	
Contributing Authors	Valérie Harscoat	Ifremer	
Reviewer(s)	Alex Vermeulen Zhiming Zhao	ICOS ERIC UvA	
Approver	Andreas Petzold	FZJ	

DELIVERY LOG

Issue	Date	Comment	Author
V 0.1	19-12-2019	Draft with template for Deliverable 9.3	V. Harscoat, T. Carval
V0.9	06-03-2020	internal review version	T. Carval
V0.9	18-03-2020	internal review	Z. Zhao A. Vermeulen

DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the Project Manager at manager@envri-fair.eu.

GLOSSARY

A relevant project glossary is included in Appendix 1. The latest version of the master list of the glossary is available at <http://doi.org/10.5281/zenodo.3465753>.

PROJECT SUMMARY

ENVRI-FAIR is the connection of the ESFRI Cluster of Environmental Research Infrastructures (ENVRI) to the European Open Science Cloud (EOSC). Participating research infrastructures (RI) of the environmental domain cover the subdomains Atmosphere, Marine, Solid Earth and Biodiversity / Ecosystems and thus the Earth system in its full complexity.

The overarching goal is that at the end of the proposed project, all participating RIs have built a set of FAIR data services which enhances the efficiency and productivity of researchers, supports innovation, enables data- and knowledge-based decisions and connects the ENVRI Cluster to the EOSC.

This goal is reached by: (1) well defined community policies and standards on all steps of the data life cycle, aligned with the wider European policies, as well as with international developments; (2) each participating RI will have sustainable, transparent and auditable data services, for each step of data life cycle, compliant to the FAIR principles. (3) the focus of the proposed work is put on the implementation of prototypes for testing pre-production services at each RI; the catalogue of prepared services is defined for each RI independently, depending on the maturity of the involved RIs; (4) the complete set of thematic data services and tools provided by the ENVRI cluster is exposed under the EOSC catalogue of services.

TABLE OF CONTENTS

1	Introduction	7
1.1	Context	7
1.2	Scope	7
2	ICOS-marine technical specification	9
2.1	Overview	9
2.1.1	Scope for the RI	9
2.1.2	Summary of the technical implementation.....	9
2.1.3	Planning for delivery.....	10
3	LifeWatch/VLIZ technical specification	10
3.1	Overview	10
3.1.1	Scope for the RI	10
3.1.2	Summary of the technical implementation.....	11
3.1.3	Planning for Delivery	11
3.2	IMIS and MDA	12
3.2.1	Description	12
3.2.2	Features	13
3.2.2.1	Feature 1 : Set up formal PID management for metadata.....	13
3.2.2.2	Feature 2 : Broaden metadata fields and ontologies	13
3.2.2.3	Feature 3 : Broaden metadata schemas.....	13
3.2.2.4	Feature 4 : Standardise m2m search webservices.....	13
3.2.2.5	Feature 5 : Improve compliance with google datasets search	14
3.2.2.6	Feature 6 : Add a m2m harvesting of IMIS	14
3.2.2.7	Feature 7 : M2M interaction with MDA	14
3.2.3	External interface	14
3.2.4	Nonfunctional requirements.....	15
3.3	LifeWatch ERIC catalogue.....	15
3.3.1	Description.....	15
3.3.2	Features	16
3.3.2.1	Feature 1 : Ecoportal catalogue of semantic objects	16
3.3.2.2	Feature 2 : LifeBlock.....	16
3.3.2.3	Feature 3 : Catalogue interoperability	17
3.3.3	External interface	17
3.3.4	Nonfunctional requirements.....	17
4	EMSO ERIC technical specification	18
4.1	Overview	18
4.1.1	Scope for the RI	18
4.1.2	Summary of the technical implementation.....	18
4.1.3	Planning for delivery.....	19
4.2	EMSO ERIC API	19
4.2.1	Description.....	19
4.2.2	Features	20
4.2.2.1	Feature 1: Authentication	20
4.2.2.2	Feature 2: Metadata	20
4.2.2.3	Feature 3: Vocabulary	21
4.2.2.4	Feature 4: Data	21
4.2.2.5	External interface	21
4.2.2.6	Nonfunctional requirements	21

4.3 ERDDAP Metadata & Data API	22
4.3.1 Description	22
4.3.2 Features	23
4.3.2.1 Feature 1: List datasets	23
4.3.2.2 Feature 2: Graph.....	23
4.3.2.3 Feature 3: WMS	23
4.3.2.4 Feature 4: Data	24
4.3.2.5 Feature 5: Display Metadata.....	24
4.3.2.6 Feature 6: Subscription.....	24
4.3.3 External interface	25
4.3.4 Nonfunctional requirements.....	25
5 Euro-Argo technical specification	27
5.1 Overview	27
5.1.1 Scope for the RI	27
5.1.2 Summary of the technical implementation.....	27
5.1.3 Planning for delivery.....	28
5.2 Argo OpenSearch API.....	28
5.2.1 Description	28
5.2.2 Features	29
5.2.2.1 Feature 1: [GET POST] /api/floats/search.....	29
5.2.2.2 Feature 2: [GET POST] /api/profiles/search	29
5.2.3 External interface	30
5.2.4 Nonfunctional requirements.....	30
5.3 Metadata API.....	31
5.3.1 Description	31
5.3.2 Features	32
5.3.2.1 Feature 1 : GET /api/floats/count	32
5.3.2.2 Feature 2 : GET /api/floats/pages?page={nb}&size={nb}	33
5.3.2.3 Feature 3 : GET /api/floats/basic/{wmo}	33
5.3.2.4 Feature 4 : GET /api/floats/{wmo}	34
5.3.2.5 Feature 5 : POST /api/floats/floats/multi-lines-count.....	35
5.3.2.6 Feature 6 : POST /api/floats/floats/multi-lines-search/pages	36
5.3.3 External interface	36
5.3.4 Nonfunctional requirements.....	37
5.4 Argo data API web service.....	37
5.4.1 Description.....	37
5.4.2 Features	38
5.4.2.1 Feature 1: GET /api/profiles	38
5.4.2.2 Feature 2: GET /api/trajectories	39
5.4.2.3 Feature 3: GET /api/timeseries.....	40
5.4.2.4 Feature 4: GET /api/profiles_timelines	41
5.4.2.5 Feature 5: GET /api/profiles_parameters	42
5.4.2.6 Feature 6: GET /api/timeseries_parameters	42
5.4.3 External interface	42
5.4.4 Nonfunctional requirements.....	43
5.5 Argo ERDDAP Metadata & Data API	43
5.5.1 Description	43
5.5.2 Features	44
5.5.2.1 Feature 1: List datasets	44
5.5.2.2 Feature 2: Graph.....	45
5.5.2.3 Feature 3: WMS	45
5.5.2.4 Feature 4: Data	45
5.5.2.5 Feature 5: Display Metadata.....	46
5.5.2.6 Feature 6: Subscription.....	46
5.5.3 External interface	46

5.5.4 Nonfunctional requirements.....	47
5.6 Argo OpenAPI (Swagger) web service	48
5.6.1 Description.....	48
5.6.2 Features	49
5.6.2.1 Feature 1: GET /api/profiles	49
5.6.2.2 Feature 2: GET /api/trajectories	50
5.6.2.3 Feature 3: GET /api/timeseries.....	51
5.6.2.4 Feature 4: GET /api/profiles_timelines	52
5.6.2.5 Feature 5: GET /api/profiles_parameters	52
5.6.2.6 Feature 6: GET /api/timeseries_parameters	52
5.6.3 External interface	53
5.6.4 Nonfunctional requirements.....	53
5.7 Argo OGC SensorThings API.....	54
5.7.1 Description.....	54
5.7.2 Features	55
5.7.2.1 Feature 1: Thing	55
5.7.2.2 Feature 2: Location.....	55
5.7.2.3 Feature 3: HistoricalLocation	56
5.7.2.4 Feature 4: Datastream.....	57
5.7.2.5 Feature 5: Sensor	58
5.7.2.6 Feature 6: ObservedProperty	59
5.7.2.7 Feature 7: Observation	59
5.7.2.8 Feature 8: FeatureOfInterest.....	61
5.7.3 External interface	62
5.7.4 Nonfunctional requirements.....	62
5.8 Argo Vocabulary web services.....	63
5.8.1 Description.....	63
5.8.2 Features	65
5.8.2.1 Feature 1: “Argo reference table 3: parameter codes” NVS collection	65
5.8.2.2 Feature 2: “All other Argo reference tables not otherwise specified“ NVS collections	65
5.8.2.3 Feature 3: “Argo configuration and technical units” NVS collection	66
5.8.2.4 Feature 4: “Argo core configuration parameter names” NVS collection	67
5.8.2.5 Feature 5: “Argo biogeochemical configuration parameter names” NVS collection	67
5.8.2.6 Feature 6: “Argo technical names” NVS collection	67
5.8.2.7 Feature 7: “Argo standard format tables” NVS collections.....	68
5.8.2.8 Feature 8: “Argo reference table 15: Codes of trajectory measurements performed within a cycle” NVS collections	68
5.8.2.9 Feature 9: Vocab Search Tool	68
5.8.2.10 Feature 10: Vocab Editor Tool	69
5.8.2.10 Feature 11: Vocab Mapping Visualisation Tool.....	69
5.8.3 External interface	70
5.8.4 Nonfunctional requirements.....	71
6 SeaDataNet technical specification	72
6.1 Overview	72
6.1.1 Scope for the RI	72
6.1.2 Summary of the technical implementation plan.....	72
6.1.3 Indicative planning for delivery	73
6.2 CDI metadata and data access API.....	74
6.2.1 Description.....	74
6.2.2 New features	74
6.2.2.1 Feature 1 : Restful API development	74
6.2.2.2 Feature 2 : Add persistency policy	75
6.2.2.3 Feature 3 : Add machine readable provenance information	75
6.2.3 External interface	75

6.2.4 Nonfunctional requirements	75
6.3 CDI SPARQL endpoint	76
6.3.1 Description	76
6.3.2 New features	77
6.3.2.1 Feature 1 : SPARQL endpoint development	77
6.3.2.2 Feature 2 : Aggregation and mapping to RDF	77
6.3.3 External interface	77
6.3.4 Nonfunctional requirements	77
6.4 Sextant data product catalogue upgrading	78
6.4.1 Description	78
6.4.2 New features	79
6.4.2.1 Feature 1 : Registration of SeaDataNet catalogue in an official registry	79
6.4.2.2 Feature 2 : Add an explicit persistency policy in metadata	79
6.4.2.3 Feature 3 : Create access to dataplots via WPS	79
6.4.2.4 Feature 4 : Upgrade metadata elements with vocabularies	79
6.4.2.5 Feature 5 : collect provenance information (QC and processing)	79
6.4.3 External interface	80
6.4.4 Nonfunctional requirements	80
7 Conclusion	81
8 References	82
9 Appendices	83
9.1 Appendix 1: Glossary	83
9.2 Appendix 2: Euro-Argo APIs examples	86

1 Introduction

1.1 Context

The ENVRI-FAIR project is engaging Research Infrastructures (RIs) in the environmental and Earth sciences domain covering the subdomains Atmosphere, Marine, Solid Earth and Biodiversity / Ecosystems. The overarching goal of ENVRI-FAIR is that all participating research infrastructures (RIs) will improve their FAIRness and become ready for connecting their data repositories and services to the European Open Science Cloud (EOSC).

WP9 has a focus on the RIs in the Marine subdomain, which is represented in ENVRI-FAIR by Euro-Argo, EMSO, and the marine component of ICOS and LifeWatch, as these RIs are listed on the ESFRI roadmap, as well as SeaDataNet as European marine data management infrastructure. The overall aim of WP9 is to analyse the status and gaps in the FAIRness of each RI and to implement within each RI the necessary actions to bridge those gaps. The latter is critical for the Marine subdomain as it will provide a coherent base for developing the integrated service oriented systems required by a broad variety of research, regulatory and operational communities.

1.2 Scope

Each RI involved in WP9 has developed an implementation plan (in D9.1) that addresses the results of their FAIRness self-analysis (D9.2), with the shared objective to improve the FAIRness at the marine subdomain level as illustrated in the following figure.

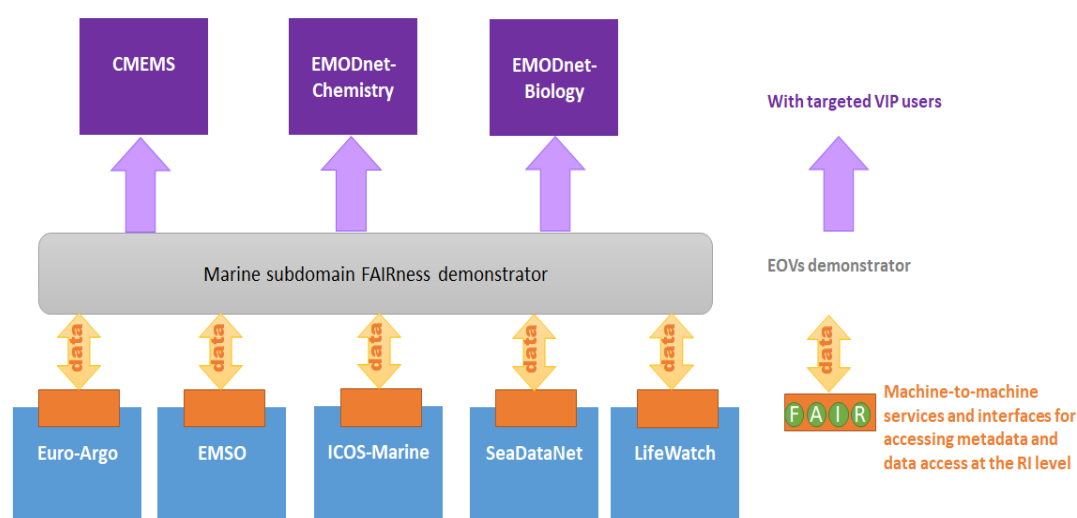


Figure 1: Marine sub-domain implementation plan

The Marine subdomain EOVs demonstrator targeted within Task 9.8 (starting end of 2020), is schematized in figure 1b. It aims to serve data files (step1) answering to requests of the VIP users termed 'Integrators' (SeaDataNet, CMEMS and EMODnet). The integrators will process data aggregation and product assessment (step 2) on the extracted data. Data files shall contain the PID to trace provenance and to allow the VIP users to provide feedback on the anomalies detected on data (step 3).

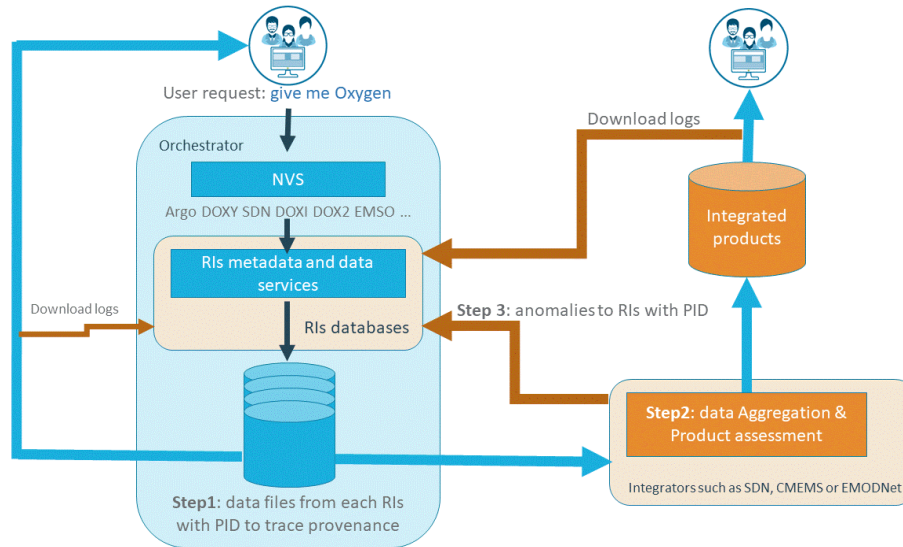


Figure 1b: General schema of the Marine subdomain EOVs demonstrator

The Marine subdomain EOVs demonstrator targeted within Task 9.8 (starting end of 2020), is schematized in figure 1b. It aims to serve data files (step1) answering to requests of the VIP users termed ‘Integrators’ (SeaDataNet, CMEMS and EMODnet). The integrators will process data aggregation and product assessment (step 2) on the extracted data. Datafiles shall contain the PID to trace provenance and to allow the VIP users to provide feedback on the anomalies detected on data (step 3).

In the present deliverable D9.3, each RI documents the technical specification of the machine-to-machine services, interfaces for accessing data and metadata to be implemented at the RI level, and detailed technical choices, and priorities planned for development.

For each RI, the planned activities for the technical implementation are among:

- The upgrade of existing machine-to-machine services and interfaces, by either improving or adding features
- The development of newly defined machine-to-machine services and interfaces
- Upgrade of shared (meta)data published via the services to increase Interoperability and Reusability

To enable the implementation of the targeted demonstrator within task 9.8, the following global pre-requirements are taken into account:

- Parameters/vocabularies : the minimum to be achieved is the request at Essential Variables level (Oxygen, Temperature, Salinity) managed in A05 in NVS and the mapping with SeaDataNet standard vocabularies (P01, P06, P07,...). the NERC Vocabulary Server (NVS) will allow to “translate” a data request “give me the EOV Oxygen” to the list of parameters in the vocabularies used by the different RIs in their data (like DOXY1, DOXY2, ... for “Oxygen”).
- Search without authentication and on the landing to download data the user enter the login if requested
- To allow the machine-to-machine extraction of the data from the RIs data, the RIs have to set up/enhance metadata and data services, such as APIs.
- A brokering of the user requests to the RI’s APIs or ERDDAP services will be set up. The broker doesn’t deal with license on data (to be done by the RI data APIs), it deals with « open and free data » in the Broker and redirects to the individual RIs for not « open and free » data. The RI API gives visibility to all data through the metadata API.
- The entire datasets containing the parameters requested have to be extracted at RI level because the context is important in the data aggregation & assessment processing by the VIP users of the EOVs demonstrator.
- At least the following provenance information has to be present in the data files: the data provider (operator of the platform, not the RI) and data aggregator (RI). The vocabulary for this provenance metadata is the EDMO codes for institutions. The citation should also to be included in the data files.

2 ICOS-marine technical specification

2.1 Overview

2.1.1 Scope for the RI

ICOS provides consistent, long-term and high-quality observations required to understand the present state and predict future behavior of the global carbon cycle and greenhouse gas emissions. ICOS RI is a distributed research infrastructure where central facilities (CFs) perform initial quality control, provide second level QC'ed data from the network and are responsible for the metadata and file content while the ICOS Carbon Portal is responsible for the technical infrastructure of long-term archival and making the (meta)data available (see Figure 1). In respect to achieving FAIRness within WP9, ICOS-marine (University of Bergen) as the CF managing the marine part of ICOS is responsible for achieving FAIRness of the metadata and data content FAIR, while the ICOS Carbon Portal (University of Lund) is responsible for the technical implementation and infrastructure.

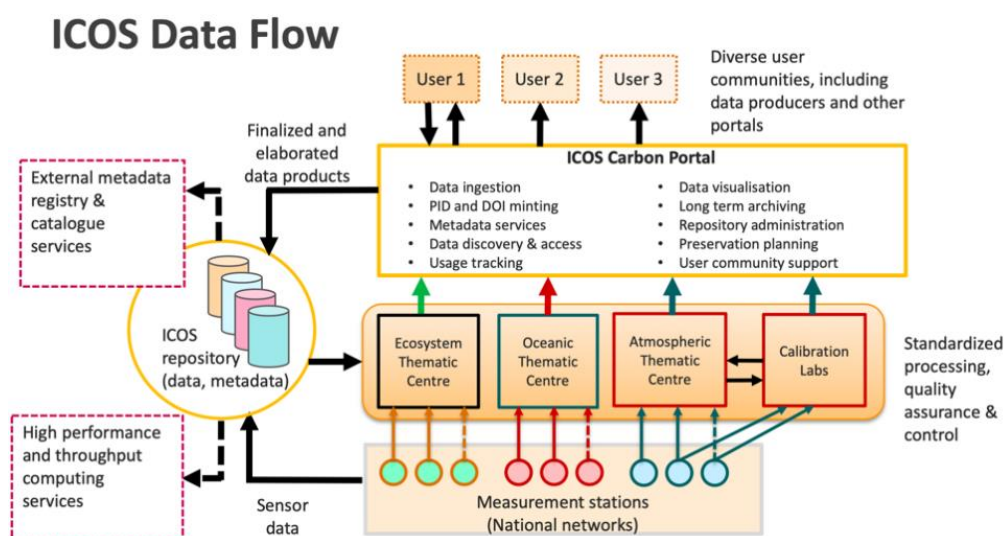


Figure 1: Scheme of the data flow with ICOS RI

2.1.2 Summary of the technical implementation

FINDABILITY	<ul style="list-style-type: none"> • Add support for keywords in metadata (present for DOIs) • Add plain text search in the search portal • Add schema.org tags in landing pages • Add support for OAI-PMH metadata exchange • Add discovery for collections in the search portal • Improve possibility to find “similar” datasets (based on a given PID/DOI)
ACCESSIBILITY	<ul style="list-style-type: none"> • Improve Data license management • Improve the way to access data object from PID • Improve, create and update the documentation on the access to data for automated workflows
INTEROPERABILITY	<ul style="list-style-type: none"> • Register the ICOS schemas in schema.org • Improve the controlled vocabularies that are not yet completely developed and published • Setup the APIs for data access and subsetting • Improve support for exporting data & metadata in other formats than ICOS domain standards

REUSABILITY

- Develop the provenance metadata at TC1 level using the templates developed in ENVRIplus and expose this through CP2
- Add optional metadata that links to measurement protocols and articles.
- Implementation of mapping to metadata schemes compliant with UN SDG (United Nations Sustainable Development Goal Methodology) methodology 14.3.1

2.1.3 Planning for delivery

Almost all improvements listed in 2.1.2 will be implemented by month 24.

In parallel, the marine community will develop the required standards that need to be added under the last point of Interoperability. After this is completed, the implementation of the metadata conversion to marine community standards will require 6 additional months.

Building support for other formats will need more detailed planning after the specifications have become clear.

Deadlines for delivery:

- M24: Content and metadata compliant at OTC and CP
- M36: Implementation of ICOS RI services

3 LifeWatch/VLIZ technical specification

3.1 Overview

3.1.1 Scope for the RI

LifeWatch ERIC is a European Infrastructure Consortium providing e-Science research facilities to scientists. As a federated network, LifeWatch is serviced by a wide range of institutes and data centres and serves a wide range of users. E-services providing access to data and tools, are a large part of what we do.

LifeWatch has various data providers and data centres that host node-specific catalogues and datasets. As one of these LifeWatch data centres, VLIZ hosts the metadata and datasets for LifeWatch Belgium, and hosts much of the marine LifeWatch data from Belgium. Numerous data systems used by LifeWatch are hosted by VLIZ: EurOBIS, EMODnet, WoRMS, and Marine Regions, all of which rely on the Marine Data Archive (MDA) and Integrated Marine Information System (IMIS). The IMIS datasets catalogue and the MDA are therefore the main focus of LifeWatch in WP9.

The LifeWatch ERIC catalogue is a new development effort for WP9 that will be released in the second half of 2020. It will be a centralised catalogue for metadata records for *all* LifeWatch data, services, and tools. Hence, this catalogue will also be a focus of the LifeWatch work in the ENVRI-FAIR project. It will be carried out mostly in WP11, but will also be in collaboration with WP9, and specifically in this report.

The focus for FAIRness improvements for the *VLIZ data systems*, as identified in D9.1 and D9.2, will be on the back-end (to improve the machine interoperability) and on metadata and data layer (to improve the completeness and standardisation of the contents), as illustrated in the diagram below. (Human) user interfaces and everything focused on the user experience is of scope for this report.

¹ TC: Thematic Centre

² CP: Carbon Portal

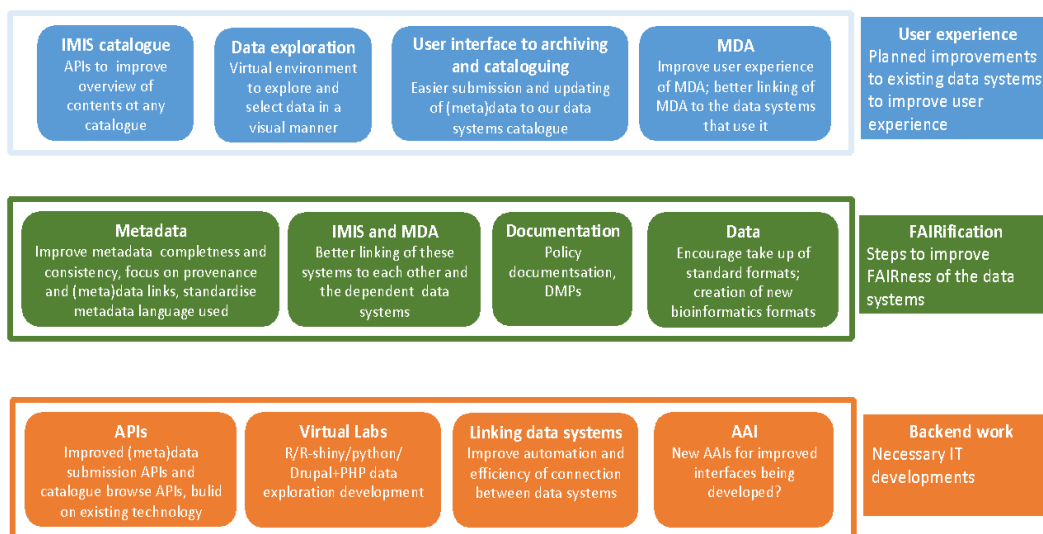


Figure 2: Focus of FAIRness actions for Lifewatch/VLIZ

The *LifeWatch ERIC metadata catalogue* is currently under construction. The catalogue will harvest only metadata records from the various LifeWatch member data providers, and from other RIs, in a machine2machine way. Most LifeWatch (meta)data are already provided in interoperable formats because they are currently also accessible via data explorers and other e-tools on www.lifewatch.eu and the individual LifeWatch country sites (e.g. www.lifewatch.be).

3.1.2 Summary of the technical implementation

The Integrated Marine Information System ([IMIS](#)) is the metadata catalogue of VLIZ that is used by LifeWatch Marine/Belgium. The Marine Data Archive is the VLIZ archive that contains much of LifeWatch Belgium data and is the starting point for all LifeWatch VLIZ dataset management. Via its link to IMIS, the archived data is made public and accessible. These services are also used by the other RIs, institutes and projects that VLIZ works with/on.

The focus of the work of VLIZ on making our services more FAIR and cloud compatible will be on IMIS and the MDA. The work to be done is detailed in Sec. 3.2. In summary, we will be: implementing more direct and standardised machine2machine interfaces to IMIS and an intermediate m2m interface to the MDA; updating and expanding the IMIS metadata schemas. We are also working on improving the completeness of the contents of our metadata records and updating the the IMIS search and submit UIs, but these activities are not part of this report.

While developing the LifeWatch ERIC catalogue is still on-going, we aim to build FAIRness in from the start. The catalogue will harvest all records from the LifeWatch node catalogues, hence machine2machine interfaces will be part of its remit from the beginning. It will also connect to LifeWatch Virtual Research Environments. A “LifeWatch EML metadata profile” is being developed to standardise the vocabularies used while accommodating the wide range of LifeWatch science. The LifeWatch EcoPortal will be used to map between all vocabularies within the metadata records that will be added to the catalogue. The details of the work to do are given in Sec. 3.3.

3.1.3 Planning for Delivery

The LifeWatch ERIC catalogue exists already but is not yet much populated.

- Work on developing a LifeWatch profile in the metadata schema (EML 2.2.0), on adding semantic resources, and on technical protocols and metadata mappings for harvesting activities, is taking place in the spring and summer of 2020.
- However, these are continuous developments that will continue through 2020 and 2021.
- Work on connecting the catalogue to a blockchain is ongoing, with a first working version expected in the autumn of 2020.

We are still in the planning stages of our work for IMIS and the MDA, and it is difficult at present to commit to specific deadlines. In addition, we have some technical questions about how D9.8 will

proceed, and the answers to these questions will have an impact on our priorities and on particular solutions we may choose for some of the features listed in Sec. 3.2. However, we can state the following:

- We will focus at first on what is necessary for the LifeWatch ERIC catalogue to be able to harvest records from IMIS (feature 6 in Sec 3.2). Discussions on this will begin in May (lockdown delay) and will start soon after (~M15 of ENVRI-FAIR).
- We are aiming to complete our improvement to the IMIS m2m searchability (feature 4), in particular to match the requirements of D9.8, by M24 of ENVRI-FAIR.
- IMIS will adopt the EML 2.2.0 metadata schema from LifeWatch ERIC, once they have completed their "LW EML profile". Modifying that profile to work for the broader IMIS database, and mapping the database to EML 2.2.0, is expected to begin in the summer 2020 (~M16) and take a year (feature 3).
- Improving the contents of our metadata records and adding new fields (e.g. provenance) is continuous work for 2020 and 2021 (feature 2).
- The other features of Sec. 3.2 are of a lower priority but our aim is for ~M30.

3.2 IMIS and MDA

3.2.1 Description

SERVICE PERSPECTIVE	<p>IMIS is a metadata catalogue for datasets, publications, institutes, persons and projects. IMIS is a data system on top of a relational database, with web and MicrosoftAccess interfaces.</p> <p>The metadata records are always distributed in html and XML, and additionally in EML schema and JSON format.</p> <p>Next to the IMIS interfaces, limited to registered users for entering records, the "metasubmit" web form allows anyone to provide metadata record information and linked datasets to the system.</p> <p>The records in IMIS can be accessed via a search interface or web services. The APIs behind the catalogue search and the metasubmit form are open access, and are also used by the projects and systems that use IMIS as their catalogue.</p> <p>Data are linked in IMIS records as PIDs (DOIs or URIs), so the data can be accessed no matter where they are actually archived.. Through DataCite VLIZ can provide DOIs for the metadata record (and hence the linked data). IMIS records without a DOI can be accessed via their (IMIS) URI instead.</p>
FEATURES	<p>To improve the FAIRness, IMIS needs to</p> <ol style="list-style-type: none"> 1. Set up a formal PID management for metadata: F 2. Broaden metadata fields and ontologies: F,I,R 3. Broaden metadata schemas: F,I 4. Standardise m2m search webservice: F,A 5. Improve compliance with google datasets search: F 6. Add a m2m harvesting of IMIS: A 7. M2M interaction with MDA: A
USER OVERVIEW	<p>IMIS is used by systems such as EurOBIS, WoRMS, Marine Regions, by RIs such as EMBRC and LifeWatch (Belgium), and by individual scientific institutes and scientists.</p>
TECHNOLOGIES	<p>PHP, MS SQL</p>
OPERATING ENVIRONMENT	<ul style="list-style-type: none"> • MS SQL Server on MS Windows Server • PHP on Apache web server
CONSTRAINTS	<p>Metadata offered in chosen metadata formats.</p> <p>Harvesting from IMIS and MDA is still under investigation.</p> <p>While adding m2m capabilities, we have to remain operational in our existing services.</p>

DOCUMENTATION	User and policy documentation to be placed on website (HTML, PDF). Human rather than machine-actionable
ASSUMPTIONS/ DEPENDENCIES	Harvester must be able to process offered metadata formats.

3.2.2 Features

3.2.2.1 Feature 1 : *Set up formal PID management for metadata*

DESCRIPTION AND PRIORITY	Set up PID management for the metadata records in IMIS: <ul style="list-style-type: none"> Formalise our policy for persistence, uniqueness, and for versioning. Add -formatted PID (URI or DOI) of the metadata record, to the metadata record itself. Ditto for the PID to the linked data.
USE CASES	None
ADDITIONAL REQUIREMENTS	Add to the policy documentation

3.2.2.2 Feature 2 : *Broaden metadata fields and ontologies*

DESCRIPTION AND PRIORITY	FAIR recommendations for new metadata fields, such as those serving provenance, will be added as necessary. Ontologies used will be updated as new incoming data have broader requirements.
USE CASES	Data types coming in now are much broader than in the past: bioinformatics data combining biological, physical, chemical, and genetic in a single data set require broader range of ontologies and semantic support in the metadata
ADDITIONAL REQUIREMENTS	Need to check recommendations on FAIR requirements on provenance metadata to come from the various EOSC/FAIR projects. Will need to match existing records to new ontologies.

3.2.2.3 Feature 3 : *Broaden metadata schemas*

DESCRIPTION AND PRIORITY	Metadata schemas provided will be broadened, potentially adding DataCite metadata schema.dopt JSON-LD. Updating to EML 2.2.0.
USE CASES	The FAIR community uses these types, EML 2.2.0 offers support for multiple ontologies in a single record. IMIS webservices will use these schemas in their results.
ADDITIONAL REQUIREMENTS	Mapping between old and new schemas for non-common fields may be necessary. Mapping from IMIS database to the metadata schema will be necessary.

3.2.2.4 Feature 4 : *Standardise m2m search webservices*

DESCRIPTION AND PRIORITY	Looking at solutions for m2m searches on IMIS. IMIS has a REST-based webservice for searching in IMIS on a range of metadata fields. We will enhance this by applying the metadata schemas (Feature 3) to the results, by adding machine-friendly features (e.g. pagination), and broadening the fields searched. The work done for Feature 6 will have some impact here, as the technical solutions for <i>searchability</i> of our catalogue may be updated as we improve its <i>harvestability</i> .
---------------------------------	---

USE CASES	ENVRI-FAIR D9.8 and the m2m search it will require. Harvesting IMIS records into the LifeWatch ERIC catalogue (this will be the first focus of this work).
ADDITIONAL REQUIREMENTS	Working together with the LifeWatch ERIC catalogue to define the methods, and hence this is a community effort. It would be useful to know more about how D9.8 will be carried out, as its technical requirements may have an impact on our work for Feature 4 and 6.

3.2.2.5 Feature 5 : *Improve compliance with google datasets search*

DESCRIPTION AND PRIORITY	Improve compliance with google dataset search to improve visibility of IMIS records via third party web-searches. Timeline: This has a low priority and will be done as time allows.
USE CASES	Users can search via google for what we have.
ADDITIONAL REQUIREMENTS	We need google to accept all the parts of bioschemas.org.

3.2.2.6 Feature 6 : *Add a m2m harvesting of IMIS*

DESCRIPTION AND PRIORITY	Making IMIS m2m harvestable will require <ul style="list-style-type: none"> • System core update to keep track of updates and changes • Enhance search webservice to account for modified-since parameter • Either applying a community-standard catalogue (including a harvesting protocol; e.g. GeoNetwork, IPT...), or extending OAI-PMH to the IMIS datasets catalogue (IMIS publications catalogue already has this).
USE CASES	M2M interoperability (metadata harvesting), e.g. from LifeWatch ERIC catalogue
ADDITIONAL REQUIREMENTS	It would be useful to know what other WP9 members are doing here, as input to our decision-making process.

3.2.2.7 Feature 7 : *M2M interaction with MDA*

DESCRIPTION AND PRIORITY	Allow programmable usage of the MDA's core functions: <ul style="list-style-type: none"> • Building a service that implements an API so users of the MDA can run in a session (post log-in) • Bulk adding/retrieving of files to folders (in-house use),
USE CASES	Direct upload of data to the MDA, e.g. from the VLIZ research vessel's data systems
ADDITIONAL REQUIREMENTS	None

3.2.3 External interface

USER INTERFACE	IMIS is accessed externally via a web page, both for the search page and the metasubmit form. UI is available to be added to other websites.
-----------------------	--

SOFTWARE INTERFACE	Limited (to the most used fields) web input interface for publications MS-Access input interface reflecting all database fields
HARDWARE INTERFACE	Web server to SQL server interactions for searches and input
COMMUNICATION INTERFACE	Communication via web requests (m2m), limited to instruction set (OAI-PMH).

3.2.4 Nonfunctional requirements

USABILITY	None
OPERATIONAL	<ul style="list-style-type: none"> • MS SQL Server on MS Windows Server • PHP on Apache web server
PERFORMANCE	Time-out settings set to support quick responses
SECURITY	The VLIZ servers and network are set up redundantly, secured and actively monitored. Backups are taken on a daily basis (GFS set up), with copies on several (geographically separated) locations
OTHER REQUIREMENTS	None

3.3 LifeWatch ERIC catalogue

3.3.1 Description

SERVICE PERSPECTIVE	<p>The LifeWatch ERIC catalogue will provide metadata records for all LifeWatch data objects (data, services, e-tools). The catalogue will harvest records from the individual LifeWatch data centres, as well as from other RIs and individuals.</p> <p>The catalogue will be based on Geonetwork, and will store metadata records for data, services, and e-tools. For metadata records from the LW nodes, the datasets will be archived by those LW nodes, i.e. the data owners. For records provided by others, the datasets can also be archived by LW.</p> <p>A metadata record can be created via a direct entry form (web-based). The harvesting from other catalogues will be done via a m2m interface (both still under development).</p> <p>The catalogue's search functionality will provide semantic support via the EcoPortal (still under development).</p>
FEATURES	<p>List main features with brief description.</p> <p>These are the unique features of this catalogue that will focus on machine to machine FAIRness</p> <ol style="list-style-type: none"> 1. EcoPortal (http://ecoportal.lifewatchitaly.eu/): a catalogue of semantic objects : F,I,R 2. LifeBlock (a blockchain): to connect the catalogue to LifeWatch workflows and VREs, and to track provenance throughout the data/software use-cycle: R 3. Catalogue interoperability: to allow m2m harvesting to and from the catalogue: A,I

USER OVERVIEW	<p>The LifeWatch data centres and institutes will archive all their metadata records in this catalogue.</p> <p>LifeWatch virtual research environments will access to and from this catalogue, via a blockchain system (LifeBlock).</p> <p>Scientists, research organisations, and industry will use the catalogue to access the data produced by LifeWatch (e.g. tracking data, long-term monitoring, surveys, etc), as they do already with the individual LW node catalogues.</p> <p>We will also interact with other RIs to share records.</p>
TECHNOLOGIES	Java, Geonetwork and NCBO technologies, Ruby on Rails, BlockChain, relational database and triple stores
OPERATING ENVIRONMENT	Docker (and also see above)
CONSTRAINTS	None
DOCUMENTATION	User and policy documentation to be placed on website (HTML, PDF) Human rather than machine-actionable (although a machine-actional DMP is a possibility)
ASSUMPTIONS/DEPENDENCIES	None

3.3.2 Features

3.3.2.1 Feature 1 : *Ecoportal catalogue of semantic objects*

DESCRIPTION AND PRIORITY	<p>The goal is to collect all the semantic objects used by LifeWatch and the ecological community.</p> <p>This is an important tool for the findability of terms and will be useful for the mapping between/alignment of the different approaches from the data providers.</p> <p>EcoPortal exists already, but we need to populate it with new semantic resources, and to transfer and publish them in the catalogue.</p> <p>This is a continuous deployment; a new version will be released every few months.</p>
USE CASES	<p>To map between an ontology used by one data provider and a one used by the ERIC catalogue.</p> <p>To allow use of multiple controlled vocabularies in the catalogue search function.</p> <p>To be used in the metadata-population process.</p>
ADDITIONAL REQUIREMENTS	None

3.3.2.2 Feature 2 : *LifeBlock*

DESCRIPTION AND PRIORITY	<p>LifeBlock is a blockchain technology. It will allow the tracking of the use of data objects through their entire use cycle, thus ensuring accountability and provenance. Because it also replicates data and allows for versioning, it will also provide security against data loss. It will also be used for controlling access to data objects where necessary.</p> <p>The LifeBlock will need to connect to the catalogue and the VREs, and its development is therefore linked to these. Aiming for a working version in Oct. 2020.</p>
USE CASES	Data are taken from LW ERIC catalogue, worked on in a LW VRE, and then read back: full provenance information is carried by the LifeBlock

ADDITIONAL REQUIREMENTS	None
--------------------------------	------

3.3.2.3 Feature 3 : *Catalogue interoperability*

DESCRIPTION AND PRIORITY	<p>The LifeWatch ERIC catalogue will need to harvest from its distributed data providers, will need to ingest metadata records created by individuals, and will need to connect to the LifeBlock to create metadata records. It also needs to harvest from and to external RIs.</p> <p>It will use OAI-PMH, Restful APIs, and other technologies.</p> <p>Version 1 of the catalogue exists. We have started discussions with VLIZ on harvesting from IMIS. This has been delayed by the lockdowns but will continue in May.</p> <p>A next version of the catalogue with a more mature LW EML profile will be released in June, and testing of the harvesting protocols and metadata mapping should begin then.</p>
USE CASES	See above
ADDITIONAL REQUIREMENTS	None

3.3.3 External interface

USER INTERFACE	<p>A web based user interface is still under development.</p> <p>LifeWatch.eu will provide links to the catalogue, and will host the policy and related documentation.</p>
SOFTWARE INTERFACE	<p>The database will be populated with backoffice of Geonetwork (via the web)</p> <p>We will use Eclipse if it is necessary to modify Geonetwork</p> <p>Harvesting will probably be done using Restful API or other m2m interfaces: these are currently under development</p>
HARDWARE INTERFACE	None
COMMUNICATION INTERFACE	LifeWatch helpdesk system (https://www.lifewatch.eu/web/guest/help-desk)

3.3.4 Nonfunctional requirements

USABILITY	Useability, e.g. the UI, is done via Geonetwork
OPERATIONAL	None
PERFORMANCE	This is done via Geonetwork.
SECURITY	<p>Identify external policies and regulations impacting safety requirements.</p> <p>LifeBlock will handle security with respect to AAAI, data replication, and use tracking</p>
OTHER REQUIREMENTS	None

4 EMSO ERIC technical specification

4.1 Overview

4.1.1 Scope for the RI

EMSO ERIC's implementation plan for adopting FAIR (Findable, Accessible, Interoperable, Reusable) practices follows standard engineering practices and focuses on efficiency for delivering the targeted EMSO ERIC's deliverables in WP9 of the ENVRI-FAIR H2020 project.

EMSO ERIC's activities in ENVRI-FAIR WP9 are coordinated by EMSO ERIC Central Management Office (CMO); however, the EMSO ERIC data service group, which includes participants from all EMSO ERIC regional facilities, will participate in the implementation plan described in this document. The implementation plan addresses multiple objectives, including bridging the current gaps for the adoption of FAIR principles in EMSO ERIC, which requires significant developments and enhancements as opposed to other RIs. A key issue for achieving the desired level of adoption of FAIR principles is a continuous self-assessment/evaluation with respect to requirements, e.g., test comparing against collections of maturity indicators developed by GO FAIR.

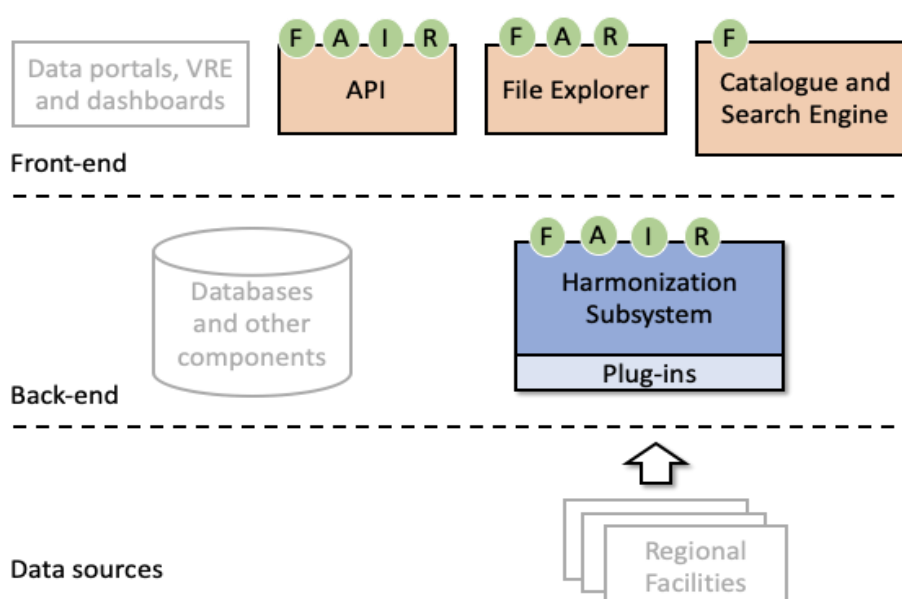


Figure 2: Illustration of focus of FAIRness activities

4.1.2 Summary of the technical implementation

FINDABILITY	<ul style="list-style-type: none"> • Deployment of software tools with data discovery capabilities that are widely used in the marine domain (e.g., ERDDAP, THREDDS) • Deployment of a (refactored) metadata catalog including both core meta-data parameters and agreed additional parameters based on an assessment of the metadata catalog against FAIR principles (emphasis on metadata enrichment). • Evaluation of metadata for discovering EMSO ERIC data using search engines. • Registration of the EMSO ERIC in research data repositories (e.g., re3data.org).
ACCESSIBILITY	<ul style="list-style-type: none"> • Deployment of a (refactored) machine-to-machine (RESTful) API. • Deployment of mechanisms for file-based discovery and access. • Implementation of mechanisms for file-based data integrity. • Development of a user database using standard tools and anonymization tools for sensitive data, when possible.

INTEROPERABILITY	<ul style="list-style-type: none"> • Integration of federated identity management capabilities. • Investigation of mechanisms for certification of repositories (e.g., CoreTrustSeal requirements). • Deployment of the harmonization subsystem based on OceanSites.
REUSABILITY	<ul style="list-style-type: none"> • Engagement with EOSC-Hub for EMSO ERIC integration in EOSC. • Initial scalability tests for EMSO-ERIC's integration in EOSC. • Engagement with a provider of persistent identifiers. • Incorporation of capabilities for PID management in the harmonization subsystem processes. • Investigation of appropriate granularity and versioning of dynamic datasets according to best practices. • Investigation of mechanisms for adding provenance information as part of the harmonization subsystem processes.

4.1.3 Planning for delivery

Harmonization subsystem, file-based access mechanism, RESTful API and engagement with providers of persistent identifiers.	M15
Refinement of harmonization subsystem and RESTful API, PID management, mechanisms for file-base data integrity, AAI mechanisms.	M18
Refinement of harmonization subsystem and other components, implementation of DAP services (e.g., ERDDAP), search engine capabilities, registration of EMSO ERIC repository.	M27
Refinement of harmonization subsystem and other components, Investigation of mechanisms for provenance data management and repository certification, documentation.	M36

4.2 EMSO ERIC API

4.2.1 Description

SERVICE PERSPECTIVE	<p>The EMSO-ERIC machine-to-machine interface has been developed to provide programmatic access to harmonized EMSO ERIC data and metadata via a Swagger-based RESTful Application Programming Interface (API) according to ENVRI-FAIR FAIRness requirements.</p> <p>This interface allows authenticated users to access EMSO ERIC data and metadata through JSON-based requests.</p> <p>The API provides ways for discovering and accessing data and metadata provided by different EMSO ERIC Regional Facilities. Data is harmonized through the harmonization subsystem using OceanSites specifications.</p>
FEATURES	<p>The EMSO ERIC RESTful API currently provides a number of endpoints as detailed in the Features chapter. The API returns the data, metadata and other related information requested by the user in JSON via HTTP:</p> <ul style="list-style-type: none"> • POST /auth/login • GET /metadata • GET /metadata/{id_metadata} • GET /vocabulary • GET /vocabulary/{vocabulary_id} • GET /data
USER OVERVIEW	<p>This API is accessible through token-based authentication. It is intended for communication between machines and building services on top of it, e.g., data</p>

TECHNOLOGIES	portals and virtual research environments.
	Main technologies and standards used: <ol style="list-style-type: none"> 1. Restful API 2. JSON
OPERATING ENVIRONMENT	The EMSO ERIC API is currently deployed in EGI resources.
CONSTRAINTS	The RESTful API requires authentication. A data policy has to be accepted.
DOCUMENTATION	A swagger is accessible. It lists the available functionalities and allows testing the different queries offered.
ASSUMPTIONS/DEPENDENCIES	N/A

4.2.2 Features

4.2.2.1 Feature 1: Authentication

DESCRIPTION AND PRIORITY	Provides a token upon successful authentication
USE CASES	Before using the EMSO ERIC API, it is needed to obtain an account. The following request provides a valid token given a valid combination of email and password. The token is refreshed every time this request is executed. <pre>curl -X POST "http://api.emso.eu/auth/login" -H "Content-Type: application/json" -d '{"email": "WRITE_HERE_YOUR_EMAIL", "password": "WRITE_HERE_YOUR_PASSWORD"}'</pre>
ADDITIONAL REQUIREMENTS	Federated authentication options to be explored according to EOSC guidelines.

4.2.2.2 Feature 2: Metadata

DESCRIPTION AND PRIORITY	Provides a list of metadata entries. Provides information related to a specific metadata entry.
USE CASES	Get the complete metadata from EMSO ERIC observatory selected by the end-user and display the meta-data in a web page. The following request provides the metadata information associated with the datasets available in the system: <pre>curl -X GET "http://api.emso.eu/metadata/" -H "Authorization: WRITE_HERE_YOUR_TOKEN"</pre> The response includes a list of elements, each of them can be identified with the parameter "id". With the following request users can obtain metadata information for id "OBSEA_2014-01-19T00:00:00Z". <pre>curl -X GET "http://api.emso.eu/metadata/OBSEA_2014-01-19T00:00:00Z" -H "Authorization: WRITE_HERE_YOUR_TOKEN" -H "Content-Type: application/json"</pre>
ADDITIONAL REQUIREMENTS	N/A

4.2.2.3 Feature 3: Vocabulary

DESCRIPTION AND PRIORITY	Provides a list of available vocabulary words. Provides the vocabulary from an ID.
USE CASES	The user/client service asks the service for the standar name, units and other relevant information of the vocabulary used. The following example provides the vocabulary of sea water temperature. curl -X GET "http://api.emso.eu/vocabulary/TEMP" -H "Authorization: WRITE_HERE_YOUR_TOKEN" -H "Content-Type: application/json"
ADDITIONAL REQUIREMENTS	N/A

4.2.2.4 Feature 4: Data

DESCRIPTION AND PRIORITY	Provides a list of data structures.
USE CASES	All datasets : the user/client service asks the service for the full list of the datasets existing on the service. List by parameters (e.g., variable, observatory) : the user/client service provides parameters existing in the datasets (e.g., variables). The service provides then a list of existing values among all the datasets that meet the specified parameters. The following example provides sea water temperature data ("parameters": ["temp"]) from OBSEA ("platform_code": "OBSEA"). This example requests only two data points ("_size": "2"). curl -X GET "http://api.emso.eu/data/" -H "Authorization: WRITE_HERE_YOUR_TOKEN" -H "Content-Type: application/json" -d '{"parameters": "temp", "platform_code": "OBSEA", "_size": "2"}'
ADDITIONAL REQUIREMENTS	N/A

4.2.2.5 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service.
SOFTWARE INTERFACE	Restful API
HARDWARE INTERFACE	N/A
COMMUNICATION INTERFACE	HTTP

4.2.2.6 Nonfunctional requirements

USABILITY	A cloud-based monitoring service checks the uptime of the service. Upon failure, an alert message is sent to the service operator.
OPERATIONAL	N/A
PERFORMANCE	The API is proposed for services such as data portals and virtual research environments.

SECURITY	The service is deployed and operated following best practises.
OTHER REQUIREMENTS	N/A

4.3 ERDDAP Metadata & Data API

4.3.1 Description

SERVICE PERSPECTIVE	The service provides metadata and data from various sources. The service gathers decentralized data in order to provide search, visualization and subset capabilities on multiple datasets.
FEATURES	<ul style="list-style-type: none"> • List datasets: provides a list of the datasets described on this server and/or others. • Graph: creates data plots or maps on selected parameters . • WMS: provides map service for gridded datasets. • Data: provides subsetting capabilities into a variety of output formats. • Display metadata: displays the metadata of the dataset and its variables. • Subscription: lets the user subscribe to an alert each time a specific dataset changes .
USER OVERVIEW	Users can be developers, data managers as well as people who want to easily discover the dataset.
TECHNOLOGIES	Java + Apache Tomcat (JVM)
OPERATING ENVIRONMENT	The service is operated on a Linux platform (Debian). The service is in a Docker container including Java, Tomcat.
CONSTRAINTS	Erddap is an open source software developed at NOAA. While it is easy to write new features, predicting release cycle is not easy.
DOCUMENTATION	<p>The documentation is split towards different actors :</p> <p>Users & developers :</p> <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/tabledap/documentation.html • https://coastwatch.pfeg.noaa.gov/erddap/griddap/documentation.html <p>System administrators :</p> <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/download/setup.html <p>Data managers :</p> <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/download/setupDatasetsXml.html
ASSUMPTIONS/DEPENDENCIES	<p>For security reasons, the infrastructure is regularly updated:</p> <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Java security patches

4.3.2 Features

4.3.2.1 Feature 1: List datasets

DESCRIPTION AND PRIORITY	The user can query for a list of the existing datasets.
USE CASES	<p>All datasets : the user/client service asks the service for the full list of the datasets existing on the server and retrieves a list with links of all the features for each dataset.</p> <p>List by dataset type : the user asks/client service for either a list of “Gridded datasets” or “Table datasets”. The server answers with the list of the datasets with a link for each of the features.</p> <p>List by attribute : the user/client service picks an attribute existing in the datasets (“variableName”, “long_name”, “standard_name”, “institution”,...). The server then provides a list of existing values among all the datasets. Once the user chose the desired value, the server provides a list of the datasets that contain the attribute and value specified by the user.</p>
ADDITIONAL REQUIREMENTS	Void

4.3.2.2 Feature 2: Graph

DESCRIPTION AND PRIORITY	For any dataset, the user can ask for plots of the data contained in the dataset.
USE CASES	<p>Case 1 – Graphical user interface The webpage presents a plot made with default values (that can be programmed by data managers when setting up the dataset). The user can change values of the parameters, for example the time range or the quality control code for a specific variable. The user clicks on “Redraw the Graph” which triggers the graph creation on the server side.</p> <p>Case 2 – Machine to machine API The same service is provided in a machine-to-machine paradigm. The client service is able to list the dataset attribute structure in order to create a query that will be sent to the API endpoint and will trigger the graph creation.</p>
ADDITIONAL REQUIREMENTS	Void

4.3.2.3 Feature 3: WMS

DESCRIPTION AND PRIORITY	For any gridded dataset, the user can ask for a colored map which shows the desired parameter value.
USE CASES	<p>Case 1 – Graphical user interface The landing webpage presents a map that shows a parameter values represented along with a color map. The user can change the parameter shown on the map. The user can zoom in/out which triggers the map creation on the server side.</p> <p>Case 2 – Machine to machine API The same service is provided in a machine-to-machine paradigm. The client service is able to create queries in a standard way (WMS standard). The client service can specify the dataset ID, parameter to color, geographical bounds, elevation in order to create the image that will be provided by the service.</p>

ADDITIONAL REQUIREMENTS	Void
--------------------------------	------

4.3.2.4 Feature 4: Data

DESCRIPTION AND PRIORITY	For any dataset, the user can ask for the data contained in the dataset in a variety of output formats.
USE CASES	<p>Case 1 – Graphical user interface The landing webpage presents a form that the user can fill in order to specify the query. The user can change values of the parameters, for example the time range or the quality control code for a specific variable. The user clicks on “Submit” which triggers the data gathering and formatting on the server side. The data can be downloaded by the user.</p> <p>Case 2 – Machine to machine API The same service is provided in a machine-to-machine paradigm. The client service is able to list the dataset attribute structure in order to create a query that will be sent to the API endpoint and will trigger the data gathering and formatting on the server side. The data can be downloaded by the user.</p>
ADDITIONAL REQUIREMENTS	Void

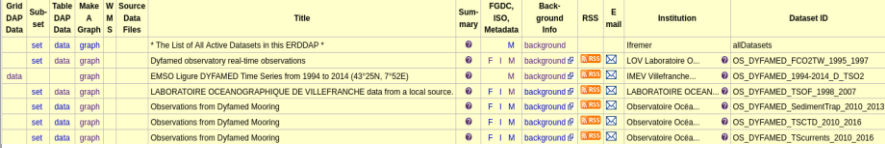
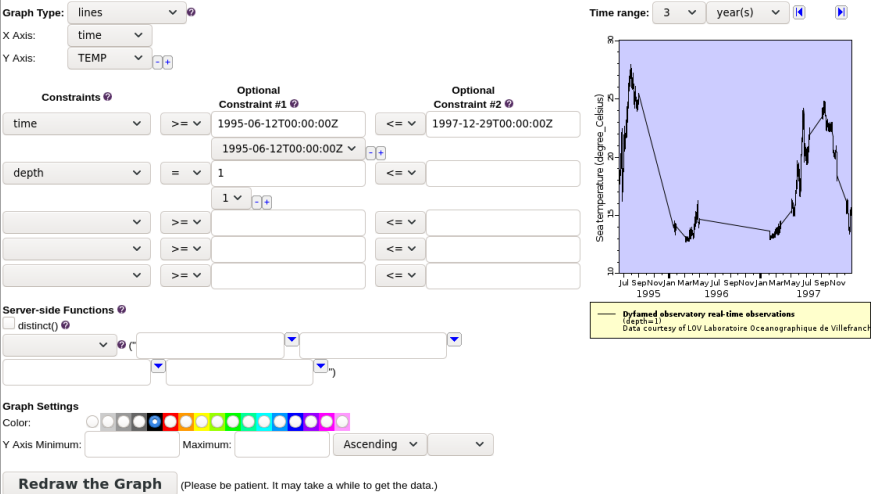
4.3.2.5 Feature 5: Display Metadata

DESCRIPTION AND PRIORITY	For any dataset, the user can retrieve the metadata describing the dataset.
USE CASES	<p>The user/client service calls the metadata URL for the desired dataset.</p> <p>Case 1 – Graphical User Interface : The server provides to the user an HTML table containing all the dataset metadata (global and variable attributes) in a human-readable way.</p> <p>Case 2 – Machine to machine API The server provides the metadata (global and variable attributes) in the format asked by the client service (can be CSV, JSON, netCDF ...)</p>
ADDITIONAL REQUIREMENTS	Void

4.3.2.6 Feature 6: Subscription

DESCRIPTION AND PRIORITY	For any dataset, the user can setup a subscription so the server can notify any change to the dataset
USE CASES	<p>The user/client service calls the subscription URL for the desired dataset. The user/client service provides the email address that will receive notifications when the dataset changes and submits the request. The server then sends an email in order for the end user to validate the subscription.</p>
ADDITIONAL REQUIREMENTS	Void

4.3.3 External interface

<p>USER INTERFACE</p>	<p>The most important parts of the webpages are presented in the form of tables. All the features for each dataset are presented in the “List datasets” feature (screenshot below)</p>  <p>Some forms cells provide drop down lists that helps the user select existing values for attributes in the dataset (screenshot below).</p> <p>ERDDAP > tabledap > Make A Graph</p> <p>Dataset Title: Dyamed observatory real-time observations B.R.S.P. Institution: LOV Laboratoire Océanographique de Villefranche (Dataset ID: OS_DYFAMED_FCO2TW_1995_1997) Range: depth = 0.0 to 1.0m, time = 1995-06-12T00:00:00Z to 1997-12-28T23:02:24Z Information: Summary License FGDC ISO 19115 Metadata Background Subset Data Access Form</p> <p>Graph Type: lines X Axis: time Y Axis: TEMP</p> <p>Time range: 3 year(s)</p>  <p>Redraw the Graph (Please be patient. It may take a while to get the data.)</p>
<p>SOFTWARE INTERFACE</p>	<p>In order to extract data and metadata from the netCDF files, Erddap relies on the Java NetCDF library which is included in the package.</p>
<p>HARDWARE INTERFACE</p>	<p>Void</p>
<p>COMMUNICATION INTERFACE</p>	<p>The server communicates with the user with the following communication standards/technologies :</p> <ul style="list-style-type: none"> • HTTP • Email • WMS (https://fr.wikipedia.org/wiki/Web_Map_Service) • OPeNDAP (https://en.wikipedia.org/wiki/OPeNDAP) <p>The user/client service can access the service with the following communication standards/technologies :</p> <ul style="list-style-type: none"> • Web Browser • Other HTTP clients : curl, python “request” module ... • Non-official python module : erddapy

4.3.4 Nonfunctional requirements

<p>USABILITY</p>	<p>A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.</p>
-------------------------	--

OPERATIONAL	<p>The service must be provided on a server that has at least 2 gigabytes of RAM.</p> <p>The Tomcat server and ERDDAP API are deployed in a docker container.</p> <p>The container is deployed on a virtual machine cluster based on VM-Ware.</p> <p>The service is deployed in the DMZ (web Internet Zone) area.</p> <p>It is requested without Internet identification.</p>
PERFORMANCE	<p>Performances depend on the filesystem which manages the NetCDF files.</p> <p>The is proposed for interactive services.</p> <p>Human users of the server expect answers to queries within a few seconds of time.</p>
SECURITY	<p>Security will be best achieved by setting up the Apache Tomcat server configuration depending on the hosting site requirements.</p> <p>The security can also be achieved by setting up a frontal web server that can implement authentication as well as SSL encryption.</p>
OTHER REQUIREMENTS	<p>The API and its Tomcat server is deployable as a docker container on EOSC infrastructure.</p>

5 Euro-Argo technical specification

5.1 Overview

5.1.1 Scope for the RI

Euro-Argo concluded from the FAIRness analysis that the system is well underway to be FAIR, but more FAIR to people than to machines. Taking this as starting point, the FAIRness improvement of Euro-Argo will focus on the back office layer, as illustrated in the next diagram.

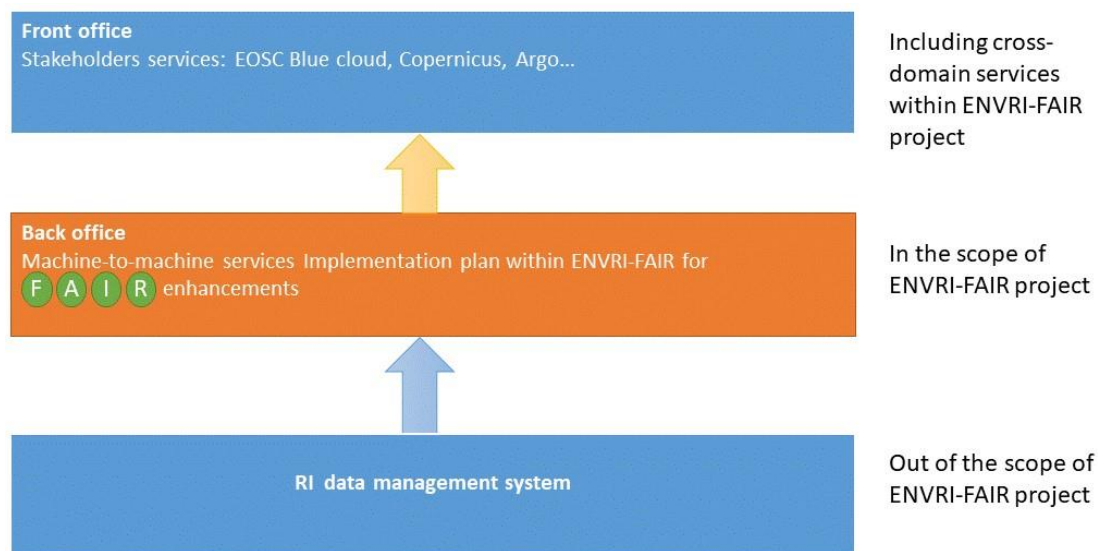


Figure 3: Illustration of focus for FAIRness activities

5.1.2 Summary of the technical implementation

FINDABILITY	Implement a search engine service to allow open queries on Argo Data
ACCESSIBILITY	<p>Implement a metadata API for machine to machine access</p> <p>Implement a data API for machine to machine access</p> <p>implement an API description service to facilitate the use of Argo APIs</p> <p>implement the major OGC services on top of data and metadata APIs: WMS for map services, SOS v3 for data queries, WPS to activate filtering and subsetting of the data processing HUB (ENVRI-FAIR VRE)</p>
INTEROPERABILITY and REUSABILITY	<p>Provide a publicly available list of standard vocabularies for Argo metadata published as linked data (NVS)</p> <p>Use existing patterns to model the data/metadata API (e.g. DCAT(1) ontology, schema.org(2) vocabulary) and serialize using json-LD (3) (json-Linked Data) format.</p> <ul style="list-style-type: none"> • https://www.w3.org/TR/vocab-dcat-2/ • https://schema.org/ • https://json-ld.org/

5.1.3 Planning for delivery

Search engine service on Argo Data	January 2021
Metadata API for machine to machine access	July 2020
Data API for machine to machine access	July 2020
ERDDAP data and metadata API	Done in 2019
API description service to facilitate the use of Argo APIs	July 2020
OGC services on top of data and metadata APIs	July 2021
WMS, SensorOfTheThings, WPS (ENVRI-FAIR VRE)	
Vocabulary server for Argo metadata	September 2020

5.2 Argo OpenSearch API

5.2.1 Description

SERVICE PERSPECTIVE	<p>The OpenSearch service is installed on top of an Elasticsearch metadata repository.</p> <p>OpenSearch is a collection of technologies that allow publishing of search results in a format suitable for syndication and aggregation. It is a way for websites and search engines to publish search results in a standard and accessible format.</p> <p>OpenSearch-OGC will be evaluated (https://www.ogc.org/standards/opensearchgeo).</p>				
FEATURES	<table> <tr> <td>/api/floats/search</td> <td>query Argo floats</td> </tr> <tr> <td>/api/profiles/search</td> <td>query Argo floats vertical profiles</td> </tr> </table>	/api/floats/search	query Argo floats	/api/profiles/search	query Argo floats vertical profiles
/api/floats/search	query Argo floats				
/api/profiles/search	query Argo floats vertical profiles				
USER OVERVIEW	<p>The API is invoked from computer applications.</p> <p>Examples:</p> <ul style="list-style-type: none"> List of floats within a bounding-box and temporal extent <p>The service is interactive: the API response time to requests should be less than a second. Pagination will be managed to keep acceptable response times.</p> <p>The API response syntax is ATOM, following OpenSearch specification.</p>				
TECHNOLOGIES	<p>The API is implemented using Java8.</p> <p>It is hosted on a Tomcat8 server (https)</p> <p>It queries an elasticsearch database.</p>				
OPERATING ENVIRONMENT	<p>The Tomcat server and API are encapsulated as a docker container.</p> <p>The container is deployed on a virtual machine cluster managed with VM-Ware.</p> <p>The service is deployed in a DMZ area.</p> <p>It is requested without Internet identification.</p>				
CONSTRAINTS	<p>The OpenSearch database is updated daily (via metadata database update).</p> <p>The web server is active 24 hours a day, 365 days a year.</p> <p>It is monitored by Nagios.</p> <p>The service must respond to at least 10 simultaneous requests in peak times.</p> <p>Response time must be less than one second.</p>				
DOCUMENTATION	<p>The Argo OpenSearch API is available online with an Internet link (https)</p> <p>The Argo OpenSearch API is documented on an OpenAPI (swagger) User Interface.</p> <p>The service may be replicated on EOSC infrastructure nodes, as a docker container.</p>				
ASSUMPTIONS/DEPENDENCIES	<p>For security reasons, the infrastructure is regularly updated:</p> <ul style="list-style-type: none"> OS patches Tomcat patches VM-Ware patches Java security patches 				

5.2.2 Features

5.2.2.1 Feature 1: [GET|POST] /api/floats/search

DESCRIPTION AND PRIORITY	<p>The API is compliant with “OGC-OpenSearch Geo and Time extensions” specification available on https://www.ogc.org/standards/opensearchgeo</p> <p>It is available in GET or POST mode. GET mode is human readable but limited in number of characters; POST mode is not readable but its JSON parameter document size is not limited.</p> <ul style="list-style-type: none"> • GET /api/floats/search?<query string> • POST /api/floats/search {JSON document } <p>Parameters</p> <ul style="list-style-type: none"> • bbox geospatial boundary (WGS 84) • datetime temporal restriction (RFC-3339) • q full text query • Sort sort results directive • page page number requested • size number of results in a page • Argo specific fields <ul style="list-style-type: none"> ○ platform_code ○ datacentre ○ owner ○ model ○ ... <p>Output An Atom document with a list of floats See example in Annex 9.2</p>
USE CASES	Query for a list of floats matching a series of criteria
ADDITIONAL REQUIREMENTS	Void

5.2.2.2 Feature 2: [GET|POST] /api/profiles/search

DESCRIPTION AND PRIORITY	<p>This feature is derived from Feature 1, for vertical profiles.</p> <p>Parameters</p> <ul style="list-style-type: none"> • bbox geospatial boundary (WGS 84) • datetime temporal restriction (RFC-3339) • q full text query • Sort sort results directive • page page number requested • size number of results in a page • Argo specific fields <ul style="list-style-type: none"> ○ platform_code ○ datacentre ○ owner ○ model ○ ... <p>Output</p>
---------------------------------	---

	An Atom document with a list of vertical profiles URLs See example in annex 9.2 .
USE CASES	Query for a list of vertical profiles
ADDITIONAL REQUIREMENTS	Void

5.2.3 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service
SOFTWARE INTERFACE	<ul style="list-style-type: none"> • Elasticsearch database • Java8 • Tomcat8
HARDWARE INTERFACE	Docker container deployed on a virtual machine, managed with VM-Ware.
COMMUNICATION INTERFACE	<p>The API is activated with https requests.</p> <p>There is no authentication.</p> <p>The communication is encrypted.</p> <p>The data transfer rate depends on the user's Internet connection.</p> <p>The server has a 10Gb internet connection.</p>

5.2.4 Nonfunctional requirements

USABILITY	A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.
OPERATIONAL	<p>The Tomcat server and API are deployed in a docker container.</p> <p>The container is activated on a virtual machine managed on a VM-Ware cluster.</p> <p>The service is deployed in the DMZ (web Internet Zone) area.</p> <p>It is requested without Internet identification.</p>
PERFORMANCE	The data API is proposed for interactive services such as Argo dashboard. Human users of the dashboard expect answers to queries within one second of time.
SECURITY	<p>The Data API is developed in a GITLAB continuous integration and deployment system.</p> <p>The GITLAB server disk has a snapshot feature: every file content change is preserved and can be restored within one month.</p> <p>The GITLAB server disk is daily archived in a remote storage. In case of a hardware failure, the archived project can be restored on another server.</p> <p>The Data API is deployed on a VM-Ware infrastructure, when a virtual machine fails, it is reactivated.</p>

OTHER REQUIREMENTS

The API and its Tomcat server can be pushed as a docker container on EOSC infrastructure.

5.3 Metadata API

In these sections, one per service or interface, present each main service or interface by completing the tables.


5.3.1 Description

<p>SERVICE PERSPECTIVE</p>	<p>The Argo metadata API provides access to metadata and technical data of Argo floats through an http REST service.</p> <p>These metadata are extracted from NetCDF Argo data files. (Figure 1-1). They are ingested in the Coriolis database (Figure 1-2). The metadata are downloaded every night with a batch and loaded in an ElasticSearch server (Figure 1-3). The Argo metadata API (Figure 1-4) provides access to the data and search functionality of this ElasticSearch engine.</p> <p>This API can feed a WEB site or any automated process with an https connection.</p> <p style="text-align: center;">Figure 4 : MetaData API architecture</p>
<p>FEATURES</p>	<p>The metadata API offers 6 types of REST requests detailed in the Features chapter. The API returns the metadata requested by the user in JSON via the https protocol :</p> <p>GET /floats/count Count the number of indexed floats</p> <p>GET /floats/pages Get paginated list of floats with only basic data</p> <p>GET /floats/basic/{wmo} Get a float basic metadata for a given wmo code</p> <p>GET /floats/{wmo} Get a float metadata with all data</p> <p>POST /floats/multi-lines-count Count the number of indexed floats for a multi-lines search</p> <p>POST /floats/multi-lines-search/pages Get paginated list of floats from multi-lines search</p>

USER OVERVIEW	This API is freely accessible without authentication. It is intended for communication between machines. It can be used to feed a WEB site for example. The applications are in interactive mode: the API response time to requests should be less than a second.
TECHNOLOGIES	This API encapsulates the search functionalities of an Elasticsearch V6 engine. Communication with the API is only performed using HTTPS requests. The API returns the requested metadata in JSON format in an HTTPS response. The API is implemented using a Java8 code. It is hosted on a Tomcat8 server (https) It queries a Cassandra database.
OPERATING ENVIRONMENT	The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine managed on a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.
CONSTRAINTS	The Cassandra database is updated daily. The web server is active 24 hours a day, 365 days a year. It is monitored by Nagios. The service must respond to at least 10 simultaneous requests The configuration of the API promotes the lowest possible latency and high output for data access.
DOCUMENTATION	A OpenAPI (swagger) is accessible. It lists the available functionalities and allows to test the different queries offered.
ASSUMPTIONS/DEPENDENCIES	for security reason, the technical infrastructure is regularly updated by: <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Elasticsearch patches • Java security patches

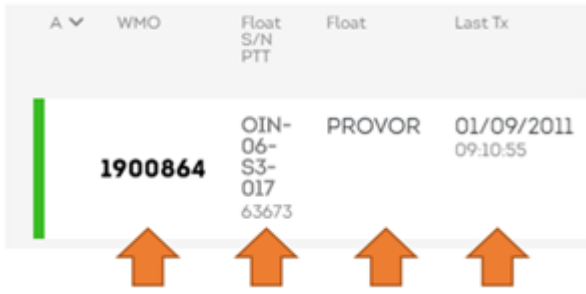
5.3.2 Features

5.4.2.1 Feature 1 : *GET /api/floats/count*

DESCRIPTION AND PRIORITY	Return the number of ARGO floats stored in the database Parameters <ul style="list-style-type: none"> • None Output <ul style="list-style-type: none"> • integer Example of output https://eoscdashboard.eu/floats/count 15958
USE CASES	Display, in a Web site, the total numbers of Argo floats available. <div style="text-align: center;">  </div> <p>Figure 2 : Display the total number of Argo floats monitored by the Argo monitoring web site</p>

ADDITIONAL REQUIREMENTS	Not applicable
--------------------------------	----------------

5.3.2.1 Feature 2 : *GET /api/floats/pages?page={nb}&size={nb}*

DESCRIPTION AND PRIORITY	<p>Get paginated list of floats with only basic metadata for each float</p> <p>Parameters</p> <ul style="list-style-type: none"> • page (integer) : page n° • size (integer) : number of floats per page <p>Output</p> <ul style="list-style-type: none"> • Response code 200 (ok), 404 (data not found) • Metadata in the response's in JSON format <p>Example of output</p> <p>https://eosc-argo-dashboard.eu/floats/pages?page=4&size=2</p> <pre style="background-color: #2e3436; color: #eeeeec; padding: 10px;"> { "wmo": "1900864", "serialInst": "OIN-06-S3-017", "serialIMEI": null, "platform": { "..."}, "sensors": [], "... }, { "wmo": "5900544", "platform": { "..."}, "... }] </pre>
USE CASES	<p>Display a result list with Argo float information in a web site</p>  <p><i>Figure 4 : Display the main information of a float in the result list in Argo monitoring web site</i></p>
ADDITIONAL REQUIREMENTS	Not applicable

5.3.2.2 Feature 3 : *GET /api/floats/basic/{wmo}*

DESCRIPTION AND PRIORITY	<p>Get the basic metadata of a specific argo float. To get the complete metadata of a float, including cycle, location, ... see Feature 4 /floats/{wmo}</p> <p>Parameters</p>
---------------------------------	--

	<ul style="list-style-type: none"> • wmo (string) <p>Output</p> <ul style="list-style-type: none"> • Response code 200 (ok), 404 (wmo not found) • If ok, metadata in the response's in JSON format <p>Example of output https://eosc-argo-dashboard.eu/floats/basic/3901909</p> <pre> { "wmo": "3901909", "serialInst": "AI2600-16FR072", "serialIMEI": null, "platform": { "code": "3901909", "name": "AI2600-16FR072", "description": "ARGO MOCCA - EU", "comment": null, ... } } </pre>
<p>USE CASES</p>	<p>Get the basic metadata of the Argo float selected by the end-user and display the meta-data in a web page.</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Not applicable</p>

5.3.2.3 Feature 4 : GET /api/floats/{wmo}

<p>DESCRIPTION AND PRIORITY</p>	<p>Get the complete set of metadata of a specific Argo float. Parameters</p> <ul style="list-style-type: none"> • wmo platform code (string) <p>Output</p> <ul style="list-style-type: none"> • Response code 200 (ok), 404 (wmo not found) • If ok, metadata in the response's in JSON format <p>Example of output https://eosc-argo-dashboard.eu/floats/3901909</p> <pre> { "wmo": "3901909", "serialInst": "AI2600-16FR072", "serialIMEI": null, "platform": { "code": "3901909", "name": "AI2600-16FR072", "description": "ARGO MOCCA - EU", "comment": null, ... }, "cycles": { "id": "112", ... } } </pre>
<p>USE CASES</p>	<p>Get the complete metadata of the Argo float selected by the end-user and display the meta-data in a web page.</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Not applicable</p>

5.3.2.4 Feature 5 : *POST /api/floats/floats/multi-lines-count*

<p>DESCRIPTION AND PRIORITY</p>	<p>Get the number of a list of floats that matches a specific query. See the multi-lines-search/pages query to get the list of index available and more information.</p> <p>Parameters The query is posted in JSON. Several index can be combined in the search</p> <p>Output</p> <ul style="list-style-type: none"> An Integer <p>Example of output</p> <pre>33 [{ "field": " piName", "searchValueType": "Text", "values": ["Birgit Klein"] }, { "field": "deploymentYear", "searchValueType": "Text", "values": ["2016"] }]</pre>
<p>USE CASES</p>	<p>A web site offers a search interface that allows the end-users to get only a selection of Argo floats. The Web site builds the corresponding API query, post the query to the API, and display the number of Argo that matches the query.</p> <div data-bbox="657 1128 1120 1382" data-label="Image"> </div> <p>Figure 5 : The Argo monitoring web site search interface</p> <div data-bbox="707 1480 1075 1599" data-label="Image"> </div> <p>Figure 6 : The Argo monitoring web site displays the number of Argo floats that match the end-user's query.</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Not applicable</p>

5.3.2.5 Feature 6 : *POST /api/floats/floats/multi-lines-search/pages*

<p>DESCRIPTION AND PRIORITY</p>	<p>Get the main metadata of a list of floats that matches a specific query.</p> <p>Parameters</p> <p>The query is posted in JSON. Several indexes can be combined in the search (See below). The response can be paginated by using the page and the size parameter (/floats/pages?page={nb}&size={nb}). The response can also be sorted (see the swagger for more information).</p> <pre>[{ "field": " piName", "searchValueType": "Text", "values": ["Birgit Klein"] }, { "field": "deploymentYear", "searchValueType": "Text", "values": ["2016"] }]</pre> <p>The following indexes are available : country, projects, InstitutionCode, platformType, basinCode, deploymentYear, platformMaker, serialIMEINumber, InstitutionDescription, cruise, dacName, status, networks,</p> <p>Output</p> <ul style="list-style-type: none"> • If ok, the metadata of the floats that match the query in JSON format <p>Example of output</p> <pre>[{ "wmo": "3901838", "serialInst": "AI2600-16FR001", "serialIMEI": null, "... "deployment": { "launchDate": "2016-08-23T09:41:00.000+0000", "year": null, "platform": "METEOR", }, "..." }, ...]</pre>
<p>USE CASES</p>	<p>See Feature 5 /floats/floats/multi-lines-count</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Not applicable</p>

5.3.3 External interface

<p>USER INTERFACE</p>	<p>There is no user interface, the API is a machine to machine service</p>
------------------------------	--

SOFTWARE INTERFACE	There is no interface apart from the 15 HTTPS queries offered by the API.
HARDWARE INTERFACE	Not applicable
COMMUNICATION INTERFACE	Not applicable

5.3.4 Nonfunctional requirements

USABILITY	A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.
OPERATIONAL	The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.
PERFORMANCE	The data API is proposed for interactive services such as Argo dashboard. Human users of the dashboard expect answers to queries within one second of time.
SECURITY	The Data API is developed in a GITLAB continuous integration and deployment system. The GITLAB server disk has a snapshot feature: every file content change is preserved and can be restored within one month. The GITLAB server disk is daily archived in a remote room. In case of a hardware failure, the archived project can be restored on another server. The Data API is deployed on a VM-Ware infrastructure, when a virtual machine fails, it is reactivated.
OTHER REQUIREMENTS	Not applicable

5.4 Argo data API web service

5.4.1 Description

SERVICE PERSPECTIVE	The Argo Data API (Application Programming Interface) is a web service on the Internet that receives requests for data and returns the data. <ul style="list-style-type: none"> • Requests are in REST format • Answers are in JSON format 	
FEATURES	/api/profiles	request for vertical profiles data
	/api/trajectories	request for trajectory data
	/api/timeseries	request for timeseries data
	/api/profiles_timelines	request for timeline (count of observation per day for a given platform)
	/api/profiles_parameters	list of parameters on a vertical profile
	/api/timeseries_parameters	list of parameters on a timeseries

USER OVERVIEW	<p>The API is invoked from computer applications. Examples:</p> <ul style="list-style-type: none"> • Data display on graphs • Data file formatting (csv, NetCDF, ...) • Calculation on the data (means, heat contents, derived parameters) <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
TECHNOLOGIES	<p>The API is a Java8 code. It is hosted on a Tomcat8 server (https) It queries a Cassandra database. Cassandra database content is encrypted. The system is configured in CP mode (CAP theorem): data consistency is guaranteed within a call.</p>
OPERATING ENVIRONMENT	<p>The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.</p>
CONSTRAINTS	<p>The Cassandra database is updated daily. The web server is active 24 hours a day, 365 days a year. It is monitored by Nagios. The service must respond to at least 10 simultaneous requests The configuration of the API promotes the lowest possible latency and high output for data access. Response time must be less than one second</p>
DOCUMENTATION	<p>The Argo data API is available online with an Internet link (https) The Argo data API is documented on an OpenAPI (swagger) User Interface. The service may be replicated on EOSC infrastructure nodes, as a docker container.</p>
ASSUMPTIONS/DEPENDENCIES	<p>For security reasons, the infrastructure is regularly updated:</p> <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Cassandra patches • Java security patches

5.4.2 Features

5.4.2.1 Feature 1: *GET /api/profiles*

DESCRIPTION AND PRIORITY	<p>Give access to profiles data, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated to physical parameter is returned, with associated quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Alternative: <code>GET /api/trajectories_goodqc</code> Identical to Feature 3 <code>api/timeseries</code> but ignores data with bad quality code (QC) Parameters</p> <ul style="list-style-type: none"> • <code>start</code> (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd).
---------------------------------	---

	<ul style="list-style-type: none"> • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • platform (string) : mandatory, platform code asked. • qc (string) : optional, qc asked. Leave blank to get all qc. • measuretype (integer) : mandatory, measure type asked. • parameter_code (integer) : mandatory, physical parameter code asked. • downsampling (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. • samplingparameter (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned <p>Output</p> <p>In the father object, these attributes are set :</p> <ul style="list-style-type: none"> • "errorcode" : set to '0' if no functional error code • "errormessage" : is set when errocode is not '0' to describe the error • "query" : the query parameters • "result" : a tab of objects. Each objects represent a specific vertical profile and have specific global attributes (date, position, associated QCs, and number of values). <p>In each object "data" attribute is a tab of tabs. Each second dimension tab include (in order) : physical parameter value, immersion level, parameter QC value, relative immersion value, and immersion QC.</p> <p>Effective data by immersion level is located in "data" attribute. It is a tab of tab. Each tab is composed by (in order) : timestamp of measure, measure value, quality code (from 0 to 9)</p> <p><u>Example of “get profiles” output in Annex 9.2</u></p>
<p>USE CASES</p>	<p>Identical to feature 1</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Void</p>

5.4.2.2 Feature 2: GET /api/trajectories

<p>DESCRIPTION AND PRIORITY</p>	<p>Give access to data from trajectories, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated to physical parameter is returned, with associated coordinates (longitude, latitude) and quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Parameters</p> <ul style="list-style-type: none"> • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • platform (string) : mandatory, platform code asked. • qc (string) : optional, qc asked. Leave blank to get all qc.
--	---

	<ul style="list-style-type: none"> • <code>measuretype</code> (integer) : mandatory, measure type asked. • <code>parameter_code</code> (integer) : mandatory, physical parameter code asked. • <code>downsampling</code> (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. • <code>samplingparameter</code> (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned <p>Output</p> <p>In the father object, these attributes are set :</p> <ul style="list-style-type: none"> • <code>"errorcode"</code> : set to '0' if no functional error code • <code>"errormessage"</code> : is set when error code is not '0' to describe the error • <code>"query"</code> : the query parameters • <code>"result"</code> : is a tab of objects. Each objects represent a specific immersion level ("<code>z_value</code>" attribute). There is also a count of data before and after downsampling (if downsampling as been selected). Effective data by immersion level is located in "<code>data</code>" attribute. It is a tab of tab. <p>Each tab is composed by (in order): timestamp of measure, measure value, lat, long, quality code (from 0 to 9)</p> <p>Example of “get trajectory” output in annex 9.2</p>
USE CASES	<p>The API is invoked from computer applications.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Data display on graphs • Data file formatting (csv, NetCDF, ...) • Calculation on the data (means, heat contents, derived parameters) <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
ADDITIONAL REQUIREMENTS	Void

5.4.2.3 Feature 3: *GET /api/timeseries*

DESCRIPTION AND PRIORITY	<p>Give access to data from timeseries, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated to physical parameter is returned, with associated quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Parameters</p> <ul style="list-style-type: none"> • <code>start</code> (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • <code>end</code> (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • <code>platform</code> (string) : mandatory, platform code asked. • <code>qc</code> (string) : optional, qc asked. Leave blank to get all qc. • <code>measuretype</code> (integer) : mandatory, measure type asked.
--------------------------	--

	<ul style="list-style-type: none"> parameter_code (integer) : mandatory, physical parameter code asked. downsampling (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. Sampling parameter (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned <p>Output In the father object, these attributes are set :</p> <ul style="list-style-type: none"> "errorcode" : set to '0' if no functional error code "errormessage" : is set when error code is not '0' to describe the error "query" : the query parameters "result" : is a tab of objects. Each objects represent a specific immersion level ("z_value" attribute). There is also a count of data before and after downsampling (if downsampling as been selected). Effective data by immersion level is located in "data" attribute. It is a tab of tab. <p>Each tab is composed by (in order): timestamp of measure, measure value, quality code (from 0 to 9)</p> <p>Example of “get timeseries” output in annex 9.2</p>
USE CASES	<p>The API is invoked from computer applications. Examples:</p> <ul style="list-style-type: none"> Data display on graphs Data file formatting (csv, NetCDF, ...) Calculation on the data (means, heat contents, derived parameters) <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
ADDITIONAL REQUIREMENTS	Void

5.4.2.4 Feature 4: GET /api/profiles_timelines

DESCRIPTION AND PRIORITY	<p>Return the number of values per day for a given platform.</p> <p>Parameters</p> <ul style="list-style-type: none"> platform (string) : mandatory, the requested platform code. <p>Output</p> <ul style="list-style-type: none"> A JSON tab filled with two attributes : "timestamp" representing the day, and "measures" representing the number of values. <p>Example</p> <pre>[{ "timestamp":1504051200000, "measures":3850 }, { "timestamp":1504483200000, "measures":3840 }, ...]</pre>
USE CASES	Identical to feature 1

ADDITIONAL REQUIREMENTS	Void
--------------------------------	------

5.4.2.5 Feature 5: *GET /api/profiles_parameters*

DESCRIPTION AND PRIORITY	<p>Return the physical parameters available for a specific profile ID.</p> <p>Parameters</p> <ul style="list-style-type: none"> • observation_id (integer) : mandatory, the profile ID requested <p>Output</p> <ul style="list-style-type: none"> • A JSON tab of associated physical parameters. <p>Example [30,35,66,67,68,69,70,71]</p>
USE CASES	Identical to feature 1
ADDITIONAL REQUIREMENTS	Void

5.4.2.6 Feature 6: *GET /api/timeseries_parameters*

DESCRIPTION AND PRIORITY	<p>Return the physical parameters available for a specific platform during a time period.</p> <p>Parameters</p> <ul style="list-style-type: none"> • platform_code (string) : mandatory, the platform code requested. • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). <p>Output</p> <ul style="list-style-type: none"> • A JSON tab of associated physical parameters. <p>Example [30,35,66,67,68,69,70,71]</p>
USE CASES	Identical to feature 1
ADDITIONAL REQUIREMENTS	Void

5.4.3 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service
SOFTWARE INTERFACE	Cassandra database Java8 Tomcat8
HARDWARE	Docker container activated on a virtual machine, managed with VM-Ware.

INTERFACE	
COMMUNICATION INTERFACE	<p>The API is activated with https requests.</p> <p>There is no identification.</p> <p>The communication is encrypted.</p> <p>The data transfer rate depends on the user's Internet connection.</p> <p>The server is connected to a 10Gb internet node.</p>

5.4.4 Nonfunctional requirements

USABILITY	<p>A Nagios agent monitors the service every 5 minutes</p> <p>Upon failure, an alert message is sent to the service owner.</p>
OPERATIONAL	<p>The Tomcat server and API are deployed in a docker container.</p> <p>The container is activated on a virtual machine in a VM-Ware cluster.</p> <p>The service is deployed in the DMZ (web Internet Zone) area.</p> <p>It is requested without Internet identification.</p>
PERFORMANCE	<p>The data API is proposed for interactive services such as Argo dashboard.</p> <p>Human users of the dashboard expect answers to queries within one second of time.</p>
SECURITY	<p>The Data API is developed in a GITLAB continuous integration and deployment system.</p> <p>The GITLAB server disk has a snapshot feature: every file content change is preserved and can be restored within one month.</p> <p>The GITLAB server disk is daily archived in a remote room. In case of a hardware failure, the archived project can be restored on another server.</p> <p>The Data API is deployed on a VM-Ware infrastructure, when a virtual machine fails, it is reactivated.</p>
OTHER REQUIREMENTS	<p>The API and its Tomcat server can be pushed as a docker container on EOSC infrastructure.</p>

5.5 Argo ERDDAP Metadata & Data API

5.5.1 Description

SERVICE PERSPECTIVE	<p>The service provides metadata and data from various sources. The service gathers decentralized data in order to provide search, visualization and subset capabilities on multiple datasets.</p>
FEATURES	<ol style="list-style-type: none"> 1. List datasets: provides a list of the datasets described on this server and/or others. 2. Graph: creates data plots or maps on selected parameters . 3. WMS: provides map service for gridded datasets. 4. Data: provides subsetting capabilities into a variety of output formats. 5. Display metadata: displays the metadata of the dataset and its variables. 6. Subscription: lets the user subscribe to an alert each time a specific dataset changes .

USER OVERVIEW	Users can be developers, data managers as well as people who want to easily discover the dataset.
TECHNOLOGIES	Java + Apache Tomcat (JVM)
OPERATING ENVIRONMENT	The service is operated on a Linux platforms (Debian). The service is in a Docker container including Java, Tomcat.
CONSTRAINTS	Erddap is an open source software developed at NOAA. While it is easy to write new features, predicting release cycle is not easy.
DOCUMENTATION	The documentation is split towards different actors : Users & developers : <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/tabledap/documentation.html • https://coastwatch.pfeg.noaa.gov/erddap/griddap/documentation.html System administrators : <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/download/setup.html Data managers : <ul style="list-style-type: none"> • https://coastwatch.pfeg.noaa.gov/erddap/download/setupDatasetsXml.html
ASSUMPTIONS/DEPENDENCIES	For security reasons, the infrastructure is regularly updated: <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Java security patches

5.5.2 Features

5.5.2.1 Feature 1: List datasets

DESCRIPTION AND PRIORITY	The user can query for a list of the existing datasets.
USE CASES	<p>All datasets : the user/client service asks the service for the full list of the datasets existing on the server and retrieves a list with links of all the features for each dataset.</p> <p>List by dataset type : the user asks/client service for either a list of “Gridded datasets” or “Table datasets”. The server answers with the list of the datasets with a link for each of the features.</p> <p>List by attribute : the user/client service picks an attribute existing in the datasets (“variableName”, “long_name”, “standard_name”, “institution”,...). The servers provides then a list of existing values among all the datasets. Once the user chose the desired value, the server provides a list of the datasets that contain the attribute and value specified by the user.</p>
ADDITIONAL REQUIREMENTS	Void

5.5.2.2 Feature 2: Graph

DESCRIPTION AND PRIORITY	For any dataset, the user can ask for plots of the data contained in the dataset.
USE CASES	<p>Case 1 – Graphical user interface The webpage presents a plot made with default values (that can be programmed by data managers when setting up the dataset). The user can change values of the parameters, for example the time range or the quality control code for a specific variable. The user clicks on “Redraw the Graph” which triggers the graph creation on the server side.</p> <p>Case 2 – Machine to machine API The same service is provided in a machine-to-machine paradigm. The client service is able to list the dataset attribute structure in order to create a query that will be sent to the API endpoint and will trigger the graph creation.</p>
ADDITIONAL REQUIREMENTS	Void

5.5.2.3 Feature 3: WMS

DESCRIPTION AND PRIORITY	For any gridded dataset, the user can ask for a colored map which shows the desired parameter value.
USE CASES	<p>Case 1 – Graphical user interface The landing webpage presents a map that shows a parameter values represented along with a color map. The user can change the parameter shown on the map. The user can zoom in/out which triggers the map creation on the server side.</p> <p>Case 2 – Machine to machine API The same service is provided in a machine-to-machine paradigm. The client service is able to create queries in a standard way (WMS standard). The client service can specify the dataset ID, parameter to color, geographical bounds, elevation in order to create the image that will be provided by the service.</p>
ADDITIONAL REQUIREMENTS	Void

5.5.2.4 Feature 4: Data

DESCRIPTION AND PRIORITY	For any dataset, the user can ask for the data contained in the dataset in a variety of output formats.
USE CASES	<p>Case 1 – Graphical user interface The landing webpage presents a form that the user can fill in order to specify the query. The user can change values of the parameters, for example the time range or the quality control code for a specific variable. The user clicks on “Submit” which triggers the data gathering and formatting</p>

	<p>on the server side. The data can be downloaded by the user.</p> <p>Case 2 – Machine to machine API</p> <p>The same service is provided in a machine-to-machine paradigm. The client service is able to list the dataset attribute structure in order to create a query that will be sent to the API endpoint and will trigger the data gathering and formatting on the server side. The data can be downloaded by the user.</p>
ADDITIONAL REQUIREMENTS	Void

5.5.2.5 Feature 5: Display Metadata

DESCRIPTION AND PRIORITY	For any dataset, the user can retrieve the metadata describing the dataset.
USE CASES	<p>The user/client service calls the metadata URL for the desired dataset.</p> <p>Case 1 – Graphical User Interface :</p> <p>The server provides to the user an HTML table containing all the dataset metadata (global and variable attributes) in a human-readable way.</p> <p>Case 2 – Machine to machine API</p> <p>The server provides the metadata (global and variable attributes) in the format asked by the client service (can be CSV, JSON, netCDF ...)</p>
ADDITIONAL REQUIREMENTS	Void

5.5.2.6 Feature 6: Subscription

DESCRIPTION AND PRIORITY	For any dataset, the user can setup a subscription so the server can notify any change to the dataset
USE CASES	<p>The user/client service calls the subscription URL for the desired dataset.</p> <p>The user/client service provides the email address that will receive notifications when the dataset changes and submits the request.</p> <p>The server then sends an email in order for the end user to validate the subscription.</p>
ADDITIONAL REQUIREMENTS	Void

5.5.3 External interface

USER INTERFACE	<p>The most important parts of the webpages is presented in the form of tables. All the features for each dataset are presented in the “List datasets” feature (screenshot below)</p> <p>Some forms cells provide drop down lists that helps the user select existing values for attributes in the dataset (screenshot below).</p>
-----------------------	--

SOFTWARE INTERFACE	In order to extract data and metadata from the netCDF files, Erddap relies on the Java NetCDF library which is included in the package.
HARDWARE INTERFACE	Void
COMMUNICATION INTERFACE	<p>The server communicates with the user with the following communication standards/technologies :</p> <ul style="list-style-type: none"> • HTTP • Email • WMS (https://fr.wikipedia.org/wiki/Web_Map_Service) • OPeNDAP (https://en.wikipedia.org/wiki/OPeNDAP) <p>The user/client service can access the service with the following communication standards/technologies :</p> <ul style="list-style-type: none"> • Web Browser • Other HTTP clients : curl, python “request” module ... • Non-official python module : erddapy

5.5.4 Nonfunctional requirements

USABILITY	A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.
OPERATIONAL	The service must be provided on a server that has at least 2 gigabytes of RAM. The Tomcat server and ERDDAP API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.
PERFORMANCE	Performances depend on the filesystem on which lie the NetCDF files. The is proposed for interactive services. Human users of the server expect answers to queries within a few seconds of time.
SECURITY	Security will be best achieved by setting up the Apache Tomcat server configuration depending on the hosting site requirements. The security can also be achieved by setting up a frontal web server that can implement authentication as well as SSL encryption.
OTHER REQUIREMENTS	The API and its Tomcat server is deployable as a docker container on EOSC infrastructure.

5.6 Argo OpenAPI (Swagger) web service

5.6.1 Description

SERVICE PERSPECTIVE	<p>The Argo OpenAPI (Swagger) is a web service to present the Argo data and metadata APIs.</p> <p>It lists and documents the APIs methods.</p> <p>It also provides a GUI (Graphic User Interface) to build and test API queries.</p>
FEATURES	<p>GET /api/profiles GET /api/trajectories GET /api/timeseries GET /api/profiles_timelines GET /api/profiles_parameters GET /api/timeseries_parameters</p>
USER OVERVIEW	<p>The API is invoked from computer applications.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Data display on graphs • Data file formatting (csv, NetCDF, ...) • Calculation on the data (means, heat contents, derived parameters) <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
TECHNOLOGIES	<p>The API is a Java8 code.</p> <p>It is hosted on a Tomcat8 server (https)</p> <p>It queries a Cassandra database.</p> <p>Cassandra database content is encrypted.</p> <p>The system is configured in CP mode (CAP theorem): data consistency is guaranteed within a call.</p>
OPERATING ENVIRONMENT	<p>The Tomcat server and API are deployed in a docker container.</p> <p>The container is activated on a virtual machine in a VM-Ware cluster.</p> <p>The service is deployed in the DMZ (web Internet Zone) area.</p> <p>It is requested without Internet identification.</p>
CONSTRAINTS	<p>The Cassandra database is updated daily.</p> <p>The web server is active 24 hours a day, 365 days a year.</p> <p>It is monitored by Nagios.</p> <p>The service must respond to at least 10 simultaneous requests</p> <p>The configuration of the API promotes the lowest possible latency and high output for data access.</p> <p>Response time must be less than one second</p>
DOCUMENTATION	<p>The Argo data API is available online with an Internet link (https)</p> <p>The Argo data API is documented on a swagger User Interface.</p> <p>The service may be replicated on EOSC infrastructure nodes, as a docker container.</p>
ASSUMPTIONS/DEPENDENCIES	<p>For security reasons, the infrastructure is regularly updated:</p> <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Cassandra patches • Java security patches

5.6.2 Features

5.6.2.1 Feature 1: GET /api/profiles

<p>DESCRIPTION AND PRIORITY</p>	<p>Give access to profiles data, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated with the physical parameter is returned, with associated quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Alternative: GET /api/trajectories_goodqc Identical to Feature 3 api/timeseries but ignores data with bad quality code (QC)</p> <p>Parameters</p> <ul style="list-style-type: none"> • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • platform (string) : mandatory, platform code asked. • qc (string) : optional, qc asked. Leave blank to get all qc. • measuretype (integer) : mandatory, measure type asked. • parameter_code (integer) : mandatory, physical parameter code asked. • downsampling (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. • Sampling parameter (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned • Output • In the father object, these attributes are set : <ul style="list-style-type: none"> • "errorcode" : set to '0' if no functional error code • "errormessage" : is set when errorcode is not '0' to describe the error • "query" : the query parameters • "result" : a tab of objects. Each object represents a specific vertical profile and has specific global attributes (date, position, associated QCs, and number of values). <p>In each object "data" attribute is a tab of tabs. Each second dimension tab includes (in order) : physical parameter value, immersion level, parameter QC value, relative immersion value, and immersion QC. Effective data by immersion level is located in "data" attribute. It is a tab of tab. Each tab is composed by (in order) : timestamp of measure, measure value, quality code (from 0 to 9)</p> <p>Example of “get profiles” output in annex 9.2</p>
<p>USE CASES</p>	<p>Identical to feature 1</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Void</p>

5.6.2.2 Feature 2: GET /api/trajectories

<p>DESCRIPTION AND PRIORITY</p>	<p>Give access to data from trajectories, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated to physical parameter is returned, with associated coordinates (longitude, latitude) and quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Parameters</p> <ul style="list-style-type: none"> • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • platform (string) : mandatory, platform code asked. • qc (string) : optional, qc asked. Leave blank to get all qc. • measuretype (integer) : mandatory, measure type asked. • parameter_code (integer) : mandatory, physical parameter code asked. • downsampling (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. • samplingparameter (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned <p>Output</p> <p>In the father object, these attributes are set :</p> <ul style="list-style-type: none"> • "errorcode" : set to '0' if no functional error code • "errormessage" : is set when error code is not '0' to describe the error • "query" : the query parameters • "result" : is a tab of objects. Each objects represent a specific immersion level ("z_value" attribute). There is also a count of data before and after downsampling (if downsampling as been selected). Effective data by immersion level is located in "data" attribute. It is a tab of tab. <p>Each tab is composed by (in order): timestamp of measure, measure value, lat, long, quality code (from 0 to 9)</p> <p>Example of “get trajectories” output in annex 9.2</p>
<p>USE CASES</p>	<p>The API is invoked from computer applications.</p> <p>Examples:</p> <p>Data display on graphs</p> <p>Data file formatting (csv, NetCDF, ...)</p> <p>Calculation on the data (means, heat contents, derived parameters)</p> <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Void</p>

5.6.2.3 Feature 3: GET /api/timeseries

<p>DESCRIPTION AND PRIORITY</p>	<p>Give access to data from timeseries, by platform, physical parameter, measure type, and time period.</p> <ul style="list-style-type: none"> • For each timestamp, a value associated to physical parameter is returned, with associated quality code (qc). • Data are grouped by immersion level and sorted by measure date ascending. • A downsampling algorithm (LTTB) can be applied before the data is returned. <p>Parameters</p> <ul style="list-style-type: none"> • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • platform (string) : mandatory, platform code asked. • qc (string) : optional, qc asked. Leave blank to get all qc. • measuretype (integer) : mandatory, measure type asked. • parameter_code (integer) : mandatory, physical parameter code asked. • downsampling (string) : optional, set the downsampling algorithm to be applied on data before the return. LTTB option is supported. Leave blank to get full data. • samplingparameter (integer) : optional, set the downsampling algorithm parameter. If LTTB is set, will determine the number of points to be returned <p>Output</p> <p>In the father object, these attributes are set :</p> <ul style="list-style-type: none"> • "errorcode" : set to '0' if no functional error code • "errormessage" : is set when error code is not '0' to describe the error • "query" : the query parameters • "result" : is a tab of objects. Each objects represent a specific immersion level ("z_value" attribute). There is also a count of data before and after downsampling (if downsampling as been selected). Effective data by immersion level is located in "data" attribute. It is a tab of tab. <p>Each tab is composed by (in order): timestamp of measure, measure value, quality code (from 0 to 9)</p> <p>Example of “get timeseries” output in Annex 9.2</p>
<p>USE CASES</p>	<p>The API is invoked from computer applications.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Data display on graphs • Data file formatting (csv, NetCDF, ...) • Calculation on the data (means, heat contents, derived parameters) <p>The applications are in interactive mode: the API response time to requests should be less than a second.</p>
<p>ADDITIONAL REQUIREMENTS</p>	<p>Void</p>

5.6.2.4 Feature 4: GET /api/profiles_timelines

DESCRIPTION AND PRIORITY	<p>Return the number of values per day for a given platform.</p> <p>Parameters platform (string) : mandatory, the requested platform code.</p> <p>Output A JSON tab filled with two attributes : "timestamp" representing the day, and "measures" representing the number of values.</p> <p>Example</p> <pre>[{ "timestamp":1504051200000, "measures":3850 }, { "timestamp":1504483200000, "measures":3840 }, ...]</pre>
USE CASES	Identical to feature 1
ADDITIONAL REQUIREMENTS	Void

5.6.2.5 Feature 5: GET /api/profiles_parameters

DESCRIPTION AND PRIORITY	<p>Return the physical parameters available for a specific profile ID.</p> <p>Parameters</p> <ul style="list-style-type: none"> • observation_id (integer) : mandatory, the profile ID requested <p>Output</p> <ul style="list-style-type: none"> • A JSON tab of associated physical parameters. <p>Example [30,35,66,67,68,69,70,71]</p>
USE CASES	Identical to feature 1
ADDITIONAL REQUIREMENTS	Void

5.6.2.6 Feature 6: GET /api/timeseries_parameters

DESCRIPTION AND PRIORITY	<p>Return physical parameters available for a specific platform during a period.</p> <p>Parameters</p> <ul style="list-style-type: none"> • platform_code (string) : mandatory, the platform code requested. • start (integer or string) : mandatory, set the beginning of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). • end (integer or string) : mandatory, set the end of time period expressed in UTC (unix timestamp or date format YYYY-mm-dd). <p>Output</p> <ul style="list-style-type: none"> • A JSON tab of associated physical parameters. <p>Example [30,35,66,67,68,69,70,71]</p>
---------------------------------	--

USE CASES	Identical to feature 1
ADDITIONAL REQUIREMENTS	Void

5.6.3 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service
SOFTWARE INTERFACE	Cassandra database Java8 Tomcat8
HARDWARE INTERFACE	Docker container activated on a virtual machine, managed with VM-Ware.
COMMUNICATION INTERFACE	The API is activated with https requests. There is no identification. The communication is encrypted. The data transfer rate depends on the user's Internet connection. The server is connected to a 10Gb internet node.

5.6.4 Nonfunctional requirements

USABILITY	A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.
OPERATIONAL	The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.
PERFORMANCE	The data API is proposed for interactive services such as Argo dashboard. Human users of the dashboard expect answers to queries within one second of time.
SECURITY	The Data API is developed in a GITLAB continuous integration and deployment system. The GITLAB server disk has a snapshot feature: every file content change is preserved and can be restored within one month. The GITLAB server disk is daily archived in a remote room. In case of a hardware failure, the archived project can be restored on another server. The Data API is deployed on a VM-Ware infrastructure, when a virtual machine fails, it is reactivated.
OTHER REQUIREMENTS	The API and its Tomcat server can be pushed as a docker container on EOSC infrastructure.

5.7 Argo OGC SensorThings API

5.7.1 Description

SERVICE PERSPECTIVE	An implementation for Argo floats of OGC SensorThings API to query sensor data on top of an O&M model.
FEATURES	<p>name "Things" url "http://a.b.c/examind/WS/sts/sts_csv/Things" Query on instruments name "Locations" url "http://a.b.c/examind/WS/sts/sts_csv/Locations" Query on positions name "Datastreams" url "http://a.b.c/examind/WS/sts/sts_csv/Datastreams" Query on observed properties name "MultiDatastreams" url "http://a.b.c/examind/WS/sts/sts_csv/MultiDatastreams" Query on combined observed properties name "Sensors" url "http://a.b.c/examind/WS/sts/sts_csv/Sensors" Query on sensors name "Observations" url "http://a.b.c/examind/WS/sts/sts_csv/Observations" Query on individual observation name "ObservedProperties" url "http://a.b.c/examind/WS/sts/sts_csv/ObservedProperties" Query available parameters name "FeaturesOfInterest" url "http://a.b.c/examind/WS/sts/sts_csv/FeaturesOfInterest" Query on regional area</p>
USER OVERVIEW	<p>The API is invoked from computer applications. Examples:</p> <ul style="list-style-type: none"> • Data access for display on graphs • Data access for download and further processing <p>The applications may be in interactive or batch mode. Interactive queries should be limited to provide a reasonable response time to requests.</p>
TECHNOLOGIES	<p>The API is a Java8 code. It is hosted on a Tomcat8 server (https) It queries the Argo data API based on Cassandra database. It is a service added on Examind server, an open source software (the development will be delivered to the community).</p>
OPERATING ENVIRONMENT	<p>The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.</p>
CONSTRAINTS	<p>The web server is active 24 hours a day, 365 days a year. It is monitored by Nagios. The service must respond to at least 10 simultaneous requests</p>
DOCUMENTATION	<p>The Argo SensorThings API will be available online with an Internet link (https) The Argo SensorThings API will be documented on an OpenAPI (swagger) User Interface. The service may be replicated on EOSC infrastructure nodes, as a docker</p>

ASSUMPTIONS/ DEPENDENCIES	container.
	<p>For security reasons, the infrastructure is regularly updated:</p> <ul style="list-style-type: none"> • Tomcat patches • VM-Ware patches • Java security patches

5.7.2 Features

5.7.2.1 Feature 1: Thing

DESCRIPTION AND PRIORITY	<p>The “Thing” is an Argo float. The OGC SensorThings API follows the ITU-T definition, i.e., with regard to the Internet of Things, a thing is an object of the physical world (physical things) or the information world (virtual things) that is capable of being identified and integrated into communication networks [ITU-T Y.2060].</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Definition</th> <th>Data type</th> <th>Multiplicity and use</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>A property provides a label for Thing entity, commonly a descriptive name.</td> <td>Character String</td> <td>One (mandatory)</td> </tr> <tr> <td>description</td> <td>This is a short description of the corresponding Thing entity.</td> <td>Character String</td> <td>One (mandatory)</td> </tr> <tr> <td>properties</td> <td>A JSON Object containing user-annotated properties as key-value pairs.</td> <td>JSON Object</td> <td>Zero-to-one</td> </tr> </tbody> </table> <p>Example of SensorThing Thing feature in annex 9.2</p>	Name	Definition	Data type	Multiplicity and use	name	A property provides a label for Thing entity, commonly a descriptive name.	Character String	One (mandatory)	description	This is a short description of the corresponding Thing entity.	Character String	One (mandatory)	properties	A JSON Object containing user-annotated properties as key-value pairs.	JSON Object	Zero-to-one
	Name	Definition	Data type	Multiplicity and use													
name	A property provides a label for Thing entity, commonly a descriptive name.	Character String	One (mandatory)														
description	This is a short description of the corresponding Thing entity.	Character String	One (mandatory)														
properties	A JSON Object containing user-annotated properties as key-value pairs.	JSON Object	Zero-to-one														
USE CASES	Query on instruments, an instrument is an Argo float																
ADDITIONAL REQUIREMENTS	Void																

5.7.2.2 Feature 2: Location

DESCRIPTION AND PRIORITY	<p>The “Location” refers to the position of observations from an Argo float. The Location entity locates the Thing or the Things it associated with. A Thing’s Location entity is defined as the last known location of the Thing. A Thing’s Location may be identical to the Thing’s Observations’ FeatureOfInterest. In the context of the IoT, the principle location of interest is usually associated with the location of the Thing, especially for in-situ sensing applications. For example, the location of interest of a wifi-connected thermostat should be the building or the room in which the smart thermostat is located. And the FeatureOfInterest of the Observations made by the thermostat (e.g., room temperature readings) should also be the building or the room. In this case, the content of the smart thermostat’s location should be the same as the content of the temperature readings’ feature of interest. However, the ultimate location of interest of a Thing is not always the location of the Thing (e.g., in the case of remote sensing). In those use cases, the content of a Thing’s Location is different from the content of the FeatureOfInterest of the Thing’s Observations. Section 7.1.4 of [OGC 10-004r3 and ISO 19156:2011] provides a detailed explanation of observation location.</p>
-----------------------------	---

	Name	Definition	Data type	Multiplicity and use
	name	A property provides a label for Location entity, commonly a descriptive name.	CharacterString	One (mandatory)
	description	The description about the Location.	CharacterString	One (mandatory)
	encoding Type	The encoding type of the Location property. Its value is one of the ValueCode enumeration.	ValueCode	One (mandatory)
	location	The location type is defined by encodingType.	Any (<i>i.e.</i> , the type is depending on the value of the encodingType)	One (mandatory)
	Example of a sensorThing location entity in annex 9.2			
USE CASES	Query on location of an Argo float			
ADDITIONAL REQUIREMENTS	Void			

5.7.2.3 Feature 3: HistoricalLocation

DESCRIPTION AND PRIORITY	The history of locations of an Argo float. A Thing's HistoricalLocation entity set provides the times of the current (<i>i.e.</i> , last known) and previous locations of the Thing.			
	Name	Definition	Data type	Multiplicity and use
	time	The time when the Thing is known at the Location.	TM_Instant (ISO-8601 Time String)	One (mandatory)
	Example of a sensorThing HistoricalLocation in annex 9.2			
USE CASES	Query on locations of an Argo float			
ADDITIONAL REQUIREMENTS	Void			

5.7.2.4 Feature 4: Datastream

DESCRIPTION AND PRIORITY	<p>The “Datastream” is a collection of an observed property of an Argo floats. A Datastream groups a collection of Observations measuring the same ObservedProperty and produced by the same Sensor.</p>			
	Name	Definition	Data type	Multiplicity and use
	name	A property provides a label for Datastream entity, commonly a descriptive name.	CharacterString	One (mandatory)
	description	The description of the Datastream entity.	CharacterString	One (mandatory)
	unitOfMeasurement	A JSON Object containing three key-value pairs. The name property presents the full name of the unitOfMeasurement; the symbol property shows the textual form of the unit symbol; and the definition contains the URI defining the unitOfMeasurement.	JSON Object	One (mandatory)
		The values of these properties SHOULD follow the Unified Code for Unit of Measure (UCUM).		Note: When a Datastream does not have a unit of measurement (e.g., a OM_TruthObservation type), the corresponding unitOfMeasurement properties SHALL have null values.
	observationType	The type of Observation (with unique result type), which is used by the service to encode observations.	ValueCode	One (mandatory)
	observedArea	The spatial bounding box of the spatial extent of all FeaturesOfInterest that belong to the Observations associated with this Datastream.	GM_Envelope (GeoJSON Polygon)	Zero-to-one (optional)
phenomenonTime	The temporal interval of the phenomenon times of all observations belonging to this Datastream.	TM_Period (ISO 8601 Time Interval)	Zero-to-one (optional)	
resultTime	The temporal interval of the result times of all observations belonging to this Datastream.	TM_Period (ISO 8601 Time Interval)	Zero-to-one (optional)	
	Example of sensorThing data stream in annex 9.2			
USE CASES	Query on observed properties of an Argo float			

ADDITIONAL REQUIREMENTS

Void

5.7.2.5 Feature 5: Sensor

DESCRIPTION AND PRIORITY	<p>An Argo float is equipped with various sensors to observe properties such as sea water temperature or sea water salinity. A Sensor is an instrument that observes a property or phenomenon with the goal of producing an estimate of the value of the property.</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Definition</th> <th>Data type</th> <th>Multiplicity and use</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>A property provides a label for Sensor entity, commonly a descriptive name.</td> <td>CharacterString</td> <td>One (mandatory)</td> </tr> <tr> <td>description</td> <td>The description of the Sensor entity.</td> <td>CharacterString</td> <td>One (mandatory)</td> </tr> <tr> <td>encoding Type</td> <td>The encoding type of the metadata property. Its value is one of the ValueCode enumeration (see Table 15 for the available ValueCode).</td> <td>ValueCode</td> <td>One (mandatory)</td> </tr> <tr> <td>metadata</td> <td>The detailed description of the Sensor or system. The metadata type is defined by encodingType.</td> <td>Any (depending on the value of the encodingType)</td> <td>One (mandatory)</td> </tr> </tbody> </table> <p>Example</p> <pre>{ "@iot.id": 1, "@iot.selfLink": "http://example.org/v1.0/Sensors(1)", "Datastreams@iot.navigationLink": "Sensors(1)/Datastreams", "name": "TMP36", "description": "TMP36 - Analog Temperature sensor", "encodingType": "application/pdf", "metadata": "http://example.org/TMP35_36_37.pdf" }</pre>				Name	Definition	Data type	Multiplicity and use	name	A property provides a label for Sensor entity, commonly a descriptive name.	CharacterString	One (mandatory)	description	The description of the Sensor entity.	CharacterString	One (mandatory)	encoding Type	The encoding type of the metadata property. Its value is one of the ValueCode enumeration (see Table 15 for the available ValueCode).	ValueCode	One (mandatory)	metadata	The detailed description of the Sensor or system. The metadata type is defined by encodingType.	Any (depending on the value of the encodingType)	One (mandatory)
	Name	Definition	Data type	Multiplicity and use																				
name	A property provides a label for Sensor entity, commonly a descriptive name.	CharacterString	One (mandatory)																					
description	The description of the Sensor entity.	CharacterString	One (mandatory)																					
encoding Type	The encoding type of the metadata property. Its value is one of the ValueCode enumeration (see Table 15 for the available ValueCode).	ValueCode	One (mandatory)																					
metadata	The detailed description of the Sensor or system. The metadata type is defined by encodingType.	Any (depending on the value of the encodingType)	One (mandatory)																					
USE CASES	Query on instruments, an instrument is an Argo float																							
ADDITIONAL REQUIREMENTS	Void																							

5.7.2.6 Feature 6: ObservedProperty

DESCRIPTION AND PRIORITY	<p>The “ObservedProperty” is a parameter observed by an Argo float. An ObservedProperty specifies the phenomenon of an Observation.</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Definition</th> <th>Data type</th> <th>Multiplicity and use</th> </tr> </thead> <tbody> <tr> <td>name</td> <td>A property provides a label for ObservedProperty entity, commonly a descriptive name.</td> <td>Character String</td> <td>One (mandatory)</td> </tr> <tr> <td>definition</td> <td>The URI of the ObservedProperty. Dereferencing this URI SHOULD result in a representation of the definition of the ObservedProperty.</td> <td>URI</td> <td>One (mandatory)</td> </tr> <tr> <td>description</td> <td>A description about the ObservedProperty.</td> <td>Character String</td> <td>One (mandatory)</td> </tr> </tbody> </table>				Name	Definition	Data type	Multiplicity and use	name	A property provides a label for ObservedProperty entity, commonly a descriptive name.	Character String	One (mandatory)	definition	The URI of the ObservedProperty. Dereferencing this URI SHOULD result in a representation of the definition of the ObservedProperty.	URI	One (mandatory)	description	A description about the ObservedProperty.	Character String	One (mandatory)
	Name	Definition	Data type	Multiplicity and use																
name	A property provides a label for ObservedProperty entity, commonly a descriptive name.	Character String	One (mandatory)																	
definition	The URI of the ObservedProperty. Dereferencing this URI SHOULD result in a representation of the definition of the ObservedProperty.	URI	One (mandatory)																	
description	A description about the ObservedProperty.	Character String	One (mandatory)																	
	<p>Example</p> <pre>{ "@iot.id": 1, "@iot.selfLink": "http://example.org/v1.0/ObservedProperties(1)", "Datastreams@iot.navigationLink": "ObservedProperties(1)/Datastreams", "description": "The dewpoint temperature is the temperature to which the air must be cooled, at constant pressure, for dew to form. As the grass and other objects near the ground cool to the dewpoint, some of the water vapor in the atmosphere condenses into liquid water on the objects.", "name": "DewPoint Temperature", "definition": "http://dbpedia.org/page/Dew_point" }</pre>																			
USE CASES	Query available parameters of an Argo float																			
ADDITIONAL REQUIREMENTS	Void																			

5.7.2.7 Feature 7: Observation

DESCRIPTION AND PRIORITY	<p>An observation from an Argo floats. An Observation is the act of measuring or otherwise determining the value of a property [OGC 10-004r3 and ISO 19156:2011]</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Definition</th> <th>Data type</th> <th>Multiplicity and use</th> </tr> </thead> <tbody> </tbody> </table>				Name	Definition	Data type	Multiplicity and use
	Name	Definition	Data type	Multiplicity and use				

	<p>phenomenonTime The time instant or period of when the Observation happens.</p> <p>Note: Many resource-constrained sensing devices do not have a clock. As a result, a client may omit phenomenonTime when POST new Observations, even though phenomenonTime is a mandatory property. When a SensorThings service receives a POST Observations without phenomenonTime, the service SHALL assign the current server time to the value of the phenomenonTime.</p> <p>result The estimated value of an ObservedProperty from the Observation.</p> <p>resultTime The time of the Observation's result was generated.</p> <p>Note: Many resource-constrained sensing devices do not have a clock. As a result, a client may omit resultTime when POST new Observations, even though resultTime is a mandatory property. When a SensorThings service receives a POST Observations without resultTime, the service SHALL assign a null value to the resultTime.</p> <p>resultQuality Describes the quality of the result.</p> <p>validTime The time period during which the result may be used.</p> <p>parameters Key-value pairs showing the environmental conditions during measurement.</p> <p>Example</p> <pre>{ "@iot.id": 1, "@iot.selfLink": "http://example.org/v1.0/Observations(1)", "FeatureOfInterest@iot.navigationLink": "Observations(1)/FeatureOfInterest", "Datastream@iot.navigationLink": "Observations(1)/Datastream", "phenomenonTime": "2014-12-31T11:59:59.00+08:00", "resultTime": "2014-12-31T11:59:59.00+08:00", "result": 21.4 }</pre>	<p>TM_Object (ISO 8601 Time string or Time Interval string (e.g., 2010-12-23T10:20:00.00-07:00 or 2010-12-23T10:20:00.00-07:00/2010-12-23T12:20:00.00-07:00))</p> <p>Any (depends on the observationType defined in the associated Datastream)</p> <p>TM_Instant (ISO 8601 Time string)</p> <p>DQ_Element</p> <p>TM_Period (ISO 8601 Time Interval string)</p> <p>NamedValues in a JSON Array</p>	<p>One (mandatory)</p> <p>One (mandatory)</p> <p>One (mandatory)</p> <p>Zero-to-many</p> <p>Zero-to-one</p> <p>Zero-to-one</p>
USE CASES	Query on individual observation of an Argo float		

ADDITIONAL REQUIREMENTS

Void

5.7.2.8 Feature 8: FeatureOfInterest

DESCRIPTION AND PRIORITY	<p>The “thing” is a collection of Argo floats. An Observation results in a value being assigned to a phenomenon. The phenomenon is a property of a feature, the latter being the FeatureOfInterest of the Observation [OGC and ISO 19156:2011]. In the context of the Internet of Things, many Observations’ FeatureOfInterest can be the Location of the Thing. For example, the FeatureOfInterest of a wifi-connect thermostat can be the Location of the thermostat (i.e., the living room where the thermostat is located in). In the case of remote sensing, the FeatureOfInterest can be the geographical area or volume that is being sensed.</p>			
	Name	Definition	Data type	Multiplicity and use
	name	A property provides a label for FeatureOfInterest entity, commonly a descriptive name.	Character String	One (mandatory)
	description	The description about the FeatureOfInterest.	Character String	One (mandatory)
	encodingType	The encoding type of the feature property. Its value is one of the ValueCode enumeration (see Table 7 for the available ValueCode).	ValueCode	One (mandatory)
	feature	The detailed description of the feature. The data type is defined by encodingType.	Any	One (mandatory)
	<p>Example</p> <pre>{ "@iot.id": 1, "@iot.selfLink": "http://example.org/v1.0/FeaturesOfInterest(1)", "Observations@iot.navigationLink": "FeaturesOfInterest(1)/Observations", "name": "Argo float Arvor 690078", "description": "This is an Argo float deployed in Atlantic ocean.", "encodingType": "application/vnd.geo+json", "feature": { "type": "Feature", "geometry": { "type": "Point", "coordinates": [-114.06,51.05] } } }</pre>			
USE CASES	Query on Argo floats observations in a regional area			

ADDITIONAL REQUIREMENTS	Void
--------------------------------	------

5.7.3 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service
SOFTWARE INTERFACE	<ul style="list-style-type: none"> • Java8 • Tomcat8 • Argo data API
HARDWARE INTERFACE	Docker container activated on a virtual machine, managed with VM-Ware.
COMMUNICATION INTERFACE	<p>The API is activated with https requests.</p> <p>There is no identification.</p> <p>The communication is encrypted.</p> <p>The data transfer rate depends on the user's Internet connection.</p> <p>The server is connected to a 10Gb internet node.</p>

5.7.4 Nonfunctional requirements

USABILITY	A Nagios agent monitors the service every 5 minutes Upon failure, an alert message is sent to the service owner.
OPERATIONAL	The Tomcat server and API are deployed in a docker container. The container is activated on a virtual machine in a VM-Ware cluster. The service is deployed in the DMZ (web Internet Zone) area. It is requested without Internet identification.
PERFORMANCE	The data API is proposed for interactive services such as Argo dashboard.
SECURITY	<p>The Data API is developed in a GITLAB continuous integration and deployment system.</p> <p>The GITLAB server disk have a snapshot feature: every file content change is preserved and can be restored within one month.</p> <p>The GITLAB server disk is daily archived in a remote room. In case of a hardware failure, the archived project can be restored on another server.</p> <p>The Data API is deployed on a VM-Ware infrastructure, when a virtual machine fails, it is reactivated.</p>
OTHER REQUIREMENTS	The API and its Tomcat server can be pushed as a docker container on EOSC infrastructure.

5.8 Argo Vocabulary web services

5.8.1 Description

SERVICE PERSPECTIVE	<p>The Argo ocean observing network consists of an array of approximately 4000 profiling floats to enable the observation of the global ocean. This has been developed and is sustained by national contributions to an international partnership of program managers, investigators, manufacturers, data managers, quality control operators, technicians and research scientists. Over the past 20 years, the Argo ocean observing system has amassed a wealth of metadata regarding float deployment, capabilities, configuration and technical status. Until now, this wealth of information has been controlled using word processing tables and spreadsheets, which would benefit from being managed in a framework that provides metadata terms and their definitions from a definitive machine-readable source.</p> <p>The intended solution is to capitalise on past investment by using the existing functionality of the NERC Vocabulary Server (NVS) to provide an Argo Vocabulary service. The NVS is hosted by the National Oceanography Centre and managed by a team of specialists at the British Oceanographic Data Centre (BODC). The NVS provides public lists of controlled terms used to describe data and metadata.</p> <p>Using standardized sets of terms ("controlled vocabularies") in metadata, and to label data, solves the problem of ambiguities associated with data markup, and enables records to be interpreted by computers. This opens data sets to a whole world of possibilities for computer aided integration, manipulation, distribution and long-term reuse.</p> <p>The NVS follows the Simple Knowledge Organisation System (SKOS) model to define, organise and classify its content. SKOS is an application of the Resource Description Framework (RDF), therefore NVS holdings can be represented on the World Wide Web and as such are machine-readable. Concepts are at the base of SKOS. They each represent an abstract entity of any kind (e.g. idea, object, event etc.), labeled with lexical strings and codes, documented through definitions, and identifiable through a Uniform Resource Identifier (URI). Concepts can be grouped into meaningful collections, aggregated into schemes, and mapped to other concepts where semantic links exist.</p> <p>The human development and maintenance of these terms is dependent on having a set of tools to support this process. The current Vocab Search Tool and Vocab Editing Tool need to be enhanced to support a growing base of users, communities and editors, whilst a Vocab Mapping Visualisation Tool needs to be developed to support the management and visualisation of mappings between concepts.</p>
FEATURES	<p>Each Argo metadata table will be presented in a unique NVS vocabulary collection and belong to one of five vocabulary collection features:</p> <ol style="list-style-type: none">1. Argo reference table for parameter codes collection;2. Argo reference tables collections (all those not specified elsewhere);3. Argo configuration and technical units collections;4. Argo core configuration parameter names collections;5. Argo biogeochemical configuration parameter names collections;6. Argo technical names collections;7. Argo standard format collections;8. Argo trajectory table collection. <p>Each feature will be managed by selected Argo governance groups, with the support of the Vocabulary Management Group and the Argo team at BODC.</p> <p>Argo metadata tables are currently listed in the Argo user's manual, cf. §3 "Reference tables" and several spreadsheets accessible from the Argo Data Management Team documentation webpage in the</p>

	<p>‘Argo data formats’ section.</p> <p>There are an additional 3 features covering the development of tools:</p> <ol style="list-style-type: none"> 9. Enhancement of the existing NVS Editor Tool to support improved human administration of the vocabulary collections; 10. Enhancement of the existing NVS Search Tool to support human improved search in identifying existing concepts and, critically, the need for new concepts; 11. Development of an NVS Mappings Visualisation Tool. <p>The prioritisation of features (1= highest priority, 5 = lowest priority) will be undertaken in two parallel streams:</p> <ul style="list-style-type: none"> • Vocabulary collection features • Vocabulary tool features <p>While these developments will facilitate access and management of the Argo vocabularies based on FAIR principles, it will also benefit the broader community.</p>
<p>USER OVERVIEW</p>	<p>Each table is human readable and machine readable, and there is a range of different types of users:</p> <ul style="list-style-type: none"> • Argo Data Management Team (ADMT) – the overall group responsible for the entire Argo data system, including Data Assembly Centres, Global Data Assembly Centres, Delayed-mode Quality Control operators and the Argo Regional Centres; • Argo Vocabulary Team – the subset of the ADMT responsible for the governance of Argo vocabulary content, including a member of the BODC Argo Team; • BODC Vocabulary Management Group – the BODC team responsible for providing technical governance for the NVS; • Argo data users; • General data users.
<p>TECHNOLOGIES</p>	<p>The Argo vocabulary collections are to be hosted on the NVS, which is managed by BODC. The technologies used are:</p> <ul style="list-style-type: none"> • ReST: The Representational State Transfer API design allows web services to be viewed as resources and can therefore be identified by their Uniform Resource Locators (URLs). • SOAP: The SOAP API design allows the exchange of structured information across computer networks following calls to web services. It relies upon XML (Extensible Markup Language) documents for passing messages. • SPARQL: a standard query language for interrogating graph databases. The SPARQL endpoint is accessible through a web interface where queries to retrieve URL lists of concept collections can be entered directly. Available output formats include XML, JSON, text, CSV and TSV.
<p>OPERATING ENVIRONMENT</p>	<p>Human users will interact with the NVS through a browser interface underpinned by the technologies listed above.</p> <p>The NVS is a service hosted at BODC; it uses a dedicated NVS server, it runs on a Linux operating system environment, and uses Oracle databases for information storage. Development and production support use a range of software including SQL database tools.</p>
<p>CONSTRAINTS</p>	<p>The range of concept attributes and mappings available is limited by the standards that are currently in use, but there is potential to adopt additional standards to extend the range of concept attributes and mappings.</p>

DOCUMENTATION	<p>Information on the NVS can be accessed through the BODC website.</p> <p>The NVS is published as Linked Data; it adheres to the following conventions:</p> <ul style="list-style-type: none"> • W3C: World Wide Web Consortium international standards; • RDF: Resource Description Framework specifications; <p>And is modeled using the following ontologies/vocabularies:</p> <ul style="list-style-type: none"> • SKOS: Simple Knowledge Organization System structure; • Dublin Core; • Provenance Authoring and Versioning ontology (PAV); • Ontology Web Language (OWL).
ASSUMPTIONS/DEPENDENCIES	<p>There are no known assumptions or dependencies that could potentially impact the technical specification, as the NVS already exists and the new features are well defined.</p>

5.8.2 Features

5.8.2.1 Feature 1: “Argo reference table 3: parameter codes” NVS collection

DESCRIPTION AND PRIORITY	<p>This NVS collection will reflect reference table 3, ‘Parameter code table’, which can be found under section 3.3 of the Argo User Manual (http://dx.doi.org/10.13155/29825).</p> <p>Thierry Carval and Mathieu Belbeoch are the appointed editors for this NVS collection.</p> <p>The priority ranking for this vocabulary collection feature is: 1.</p>
USE CASES	<p>Concepts from this collection are used for labelling variables in the Argo NetCDF files. They specify which parameter the corresponding data is associated with.</p> <p>These concepts will be mapped to the BODC Parameter Usage Vocabulary collection P01, an NVS collection containing over 40,000 terms and more than 300,000 SKOS mappings providing links to broader and related concepts; these vocabulary links are key to enabling data discovery and aggregation at a wide scale.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.2 Feature 2: “All other Argo reference tables not otherwise specified” NVS collections

DESCRIPTION AND PRIORITY	<p>These NVS collections will reflect all tables currently held solely in section 3 of the Argo User Manual (http://dx.doi.org/10.13155/29825) which are not covered by any other feature.</p> <p>Thierry Carval and Mathieu Belbeoch are the appointed editors for these NVS collections.</p> <p>The priority ranking for this vocabulary collection feature is: 1.</p>
USE CASES	<p>Concepts from these collections are used as metadata in the Argo NetCDF files. These concepts capture various information about the nature of the float including:</p> <ul style="list-style-type: none"> • File data type

	<ul style="list-style-type: none"> • Quality control flag scales • Data centres and institution codes • Classes of position accuracy • Data state indicators • History action codes • Instrument types • Positioning system • Transmission system • Binary ID of quality control test results • History steps codes • Ocean codes • Vertical sampling schemes • STATUS flags • GROUNDED flags • REPRESENTATIVE_PARK_PRESSURE_STATUS • PLATFORM_FAMILY • PLATFORM_TYPE • PLATFORM_MAKER • SENSOR • SENSOR_MAKER • SENSOR_MODEL • CONTROLLER_BOARD_TYPE_PRIMARY • BATTERY_TYPE • BATTERY_PACKS <p>Some of these terms are not yet machine readable. In addition to supporting long-term maintenance and use of these vocabularies for Argo, these concepts could be adopted for wider observing system use, or could be mapped to form other collections external to Argo to enhance semantic richness and provide a basis for cross in-domain and cross domain use.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.3 Feature 3: “Argo configuration and technical units” NVS collection

DESCRIPTION AND PRIORITY	<p>This NVS collection will reflect the list of units associated with float configuration and technical metadata. Currently, the primary location of this table is an XLSX file which can be downloaded from: Table%20Tech%20and%20Conf%20Units%20V2.3.xlsx</p> <p>Birgit Klein is the appointed editor for this NVS collection.</p> <p>The priority ranking for this vocabulary collection feature is: 1.</p>
USE CASES	<p>Concepts from this collection are used to define the units of the metadata from the Argo technical files and configuration parameters.</p> <p>Though BODC NVS collection P06 holds an extensive list of data storage units, several of those used in conjunction with Argo technical and configuration metadata are not yet present. There is thus significant scope in creating concepts to be mapped to the Argo configuration and technical parameters, either as a new collection or as additions to P06.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.4 Feature 4: “Argo core configuration parameter names” NVS collection

DESCRIPTION AND PRIORITY	<p>These NVS collections will reflect all Argo core configuration parameter names described in the Excel spreadsheet ‘Configuration parameter names, core Argo, November 13th 2018’, which can be downloaded from the http://www.argodatamgt.org/Documentation website under the ‘Argo data formats’/‘Argo metadata files’ sub-section.</p> <p>John Gilson is the appointed editor for these NVS collections.</p> <p>The priority ranking for this vocabulary collection feature is: 2.</p>
USE CASES	<p>Concepts from these collections are used as metadata in the Argo NetCDF files, in support of the Core Argo mission. These concepts define the intended float configuration, and enable the at-sea monitoring of float performance.</p> <p>There is significant scope to enhance semantic richness through a review of the definition of existing terms, and by mapping between terms. There is also the potential to introduce additional collections based on the type of float, to enable mapping between float manufacturer and Argo Data Management Team defined concepts.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.5 Feature 5: “Argo biogeochemical configuration parameter names” NVS collection

DESCRIPTION AND PRIORITY	<p>These NVS collections will reflect all Argo biogeochemical configuration parameter names described in the Excel spreadsheet ‘Configuration parameter names, BGC-Argo, July10th 2019’, which can be downloaded from the http://www.argodatamgt.org/Documentation website under the ‘Argo data formats’/‘Argo metadata files’ sub-section.</p> <p>Catherine Schmechtig is the appointed editor for these NVS collections.</p> <p>The priority ranking for this vocabulary collection feature is: 2.</p>
USE CASES	<p>Concepts from these collections are used as additional float configuration metadata in the Argo NetCDF files, in support of the Biogeochemical Argo mission. These concepts define the intended float configuration with respect to biogeochemical sensors, and enable the at-sea monitoring of float performance. There is significant scope to enhance semantic richness through a review of the definition of existing terms, and by mapping between terms.</p> <p>There is also the potential to introduce additional collections based on the type of float to enable mapping between float manufacturer and Argo Data Management Team defined concepts.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.6 Feature 6: “Argo technical names” NVS collection

DESCRIPTION AND PRIORITY	<p>These NVS collections will reflect all Argo technical names described in the documents under the ‘Argo data formats’/‘Argo user manual for technical files’ sub-section, which can be downloaded from the http://www.argodatamgt.org/Documentation website.</p> <p>Birgit Klein is the appointed editor for these NVS collections.</p> <p>The priority ranking for this vocabulary collection feature is: 2.</p>
USE CASES	Concepts from these collections are used as float technical metadata in the

	Argo NetCDF files. These concepts define the float technical status and enable its at-sea monitoring. There is significant scope to enhance semantic richness through a review of the definition of existing terms, and by mapping between terms. There is also the potential to introduce additional collections based on the type of float to enable mapping between float manufacturer and Argo Data Management Team defined concepts.
ADDITIONAL REQUIREMENTS	n/a

5.8.2.7 Feature 7: “Argo standard format tables” NVS collections

DESCRIPTION AND PRIORITY	These NVS collections will reflect the Argo standard format table, which is currently stored in a Google spreadsheet accessible from the ADMT webpage and from the Argo User Manual: http://tinyurl.com/qy7fdqc . Megan Scanderbeg and Mathieu Belbeoch are the appointed editors for these NVS collections. The priority ranking for this vocabulary collection feature is: 3 .
USE CASES	This table currently maps Argo float types to PLATFORM_TYPE_KEY in a 1:1 relationship; the latter is then mapped to STANDARD_FORMAT_ID in a 1:many relationship. Use cases and implementation of this information within the NVS still needs to be fully explored and comprehended.
ADDITIONAL REQUIREMENTS	n/a

5.8.2.8 Feature 8: “Argo reference table 15: Codes of trajectory measurements performed within a cycle” NVS collections

DESCRIPTION AND PRIORITY	These NVS collections will contain the codes of trajectory measurements performed within a cycle. These are currently in the table under section 3.15 of the Argo User Manual (http://dx.doi.org/10.13155/29825), and the Argo Trajectory Measurement Code Tables document (https://archimer.ifremer.fr/doc/00187/29824/66481.pdf) which can be found in the Argo data management website http://www.argodatamgt.org/Documentation under ‘Cookbooks, core-Argo’. An editor from the ADMT community has not been appointed yet. The priority ranking for this vocabulary collection feature is: 3 .
USE CASES	Concepts from these collections are used as metadata in the Argo NetCDF trajectory files. The concepts describe measurement timing and position relative to the float’s cycle through a series of codes.
ADDITIONAL REQUIREMENTS	n/a

5.8.2.9 Feature 9: Vocab Search Tool

DESCRIPTION AND PRIORITY	Improvement of the existing Vocabulary Search Tool https://www.bodc.ac.uk/resources/vocabularies/vocabulary_se
---------------------------------	---

	<p>arch/, such as:</p> <ol style="list-style-type: none"> 1. Simplification of the user interface: intuitive design, standard search features; 2. Quick access to collections based on governance or customised groupings; 3. Enhancement of free text search options and flexibility; 4. Simplification of the display output, with customisation enabled e.g. choice of format, fields, etc. <p>The priority ranking for this tool feature is: 1</p>
USE CASES	<p>The current content of vocabularies, including concept mappings, is used by both humans and machines; however, it is humans that must ultimately impart meaning to new terms and understand the meaning of existing terms. The human user can currently access the Vocabulary Search Tool to search for collections and concepts, and within collections for concepts; however, the tool in its current form requires improvement to enhance usability (particularly by unfamiliar users), to more readily expose all of the available semantic richness, and to provide more options for display and export in support of vocabulary reuse and development.</p>
ADDITIONAL REQUIREMENTS	n/a

5.9.2.10 Feature 10: Vocab Editor Tool

DESCRIPTION AND PRIORITY	<p>Improve existing Vocabulary Editing Tool (including mapping management) https://www.bodc.ac.uk/resources/vocabularies/vocabulary_editor/, such as:</p> <ol style="list-style-type: none"> 1. Add automated checks and improve user feedback; 2. Harmonise submission formats; 3. Enable user to review submission before committing; 4. Improve back-office tooling and workflow. <p>The priority ranking for this tool feature is: 2</p>
USE CASES	<p>Vocabularies need to be developed in a consistent manner to ensure that their semantic meaning is retained over long time periods. This is fundamental so that the NVS may grow in line with the evolving needs of the communities it serves without compromising the quality and integrity of its holdings. The current capabilities of the Vocabulary Editing Tool enable individuals with the role of vocabulary editor to administer the vocabularies they are responsible for. However, as the number of vocabularies, concepts and editors expand, the tool will need to be enhanced to provide the functionality to support consistent and effective decision-making.</p>
ADDITIONAL REQUIREMENTS	n/a

5.8.2.10 Feature 11: Vocab Mapping Visualisation Tool

DESCRIPTION AND PRIORITY	<p>Creation or adoption of a visualisation tool to help the management and development of mappings between different vocabulary concepts:</p> <ol style="list-style-type: none"> 1. Intuitive tool to explore and visualise links between concepts in e.g. Argo vocabularies and other resources; 2. Explore the possibility of linking this tool with the Vocabulary editing tool for mapping management.
---------------------------------	--

	The priority ranking for this tool feature is: 3
USE CASES	Vocabulary users and editors need to be able to understand current and potential future mappings for creating links between concepts. There are currently no intuitive tools in place, and users are reliant on plain text links within the definition of an individual concept. Improving the available tools for this purpose has the potential to enhance the semantic richness of the relationship between concepts by significantly increasing the number of mappings available; this will in turn facilitate domain and across domain use cases.
ADDITIONAL REQUIREMENTS	n/a

5.8.3 External interface

USER INTERFACE	<ul style="list-style-type: none"> • Vocabulary Search: human-readable web interface to search and access NVS collections within the whole catalogue. No access control. • NVS SPARQL endpoint: human and machine readable web interface providing access to the entire NVS catalogue through SPARQL queries, and outputting a range of selectable formats including ASCII, JSON, XML, CSV and TSV. No access control. • Vocabulary Editor: human interface allowing governance groups and representatives to manage concepts and mappings that fall under their remit. Access control in place. • NVS Linked Data REST API: operates using content negotiation, thus serving different versions of a resource at the same URI to allow user agents to specify which version fits their capabilities best. In practice, the Linked Data REST API serves html content to a human user (browser) and RDF/XML to the machine. • NVS Github repositories: open discussions, term requests and reporting of issues can be made via the NVS Github repositories. Using github empowers the transparency of governance and opens up discussions to a wider community by using a centralised open space. • SeaDataNet community tools: a set of external tools for NVS were developed by SeaDataNet and can be found here. • FAIRsharing.org: holds more information about the NVS, and can be accessed through this link. <p>The NVS content is updated twice a day.</p>
SOFTWARE INTERFACE	<p>The current NERC Vocabulary Server version is 2.0. NVS content is stored in BODC's Oracle database and is made publicly available via web services namely SOAP, REST and SPARQL. As stated above, the REST interface follows the Linked Data principles and applies content negotiation to serve its content differently to humans and machines.</p> <p>NVS functionality is enhanced with a set of tools for searching through and editing of vocabularies with the NVS Search and Editor respectively.</p>
HARDWARE INTERFACE	N/A
COMMUNICATION INTERFACE	<p>The Vocabulary Editor tool requires user identification to create/edit concepts and create/edit mappings for collections owned by the user.</p> <p>The majority of the remaining NVS web services and applications can be used without login.</p>

5.8.4 Nonfunctional requirements

USABILITY	The NVS is already an operational service which serves a large community of globally distributed users. It currently receives an average of ~10,000 calls per day through its different web services.
OPERATIONAL	The NVS will be provided through a dedicated virtual machine with redundant processing and database capabilities.
PERFORMANCE	<p>The NVS underpins the SeaDataNet data infrastructure and services. It is continuously monitored for its performance, reliability and availability, both internally at BODC but also by its main user communities (see e.g. https://sdc.ui.argo.gmet.gr/sdc/dashboard/Critical/VOCABULARY).</p> <p>The Argo project vocabulary services will benefit from the same level of performance monitoring. It is worth mentioning that our current user best practices of consuming NVS webservices is caching.</p>
SECURITY	The content of the NVS does not need to be protected as it is freely available for use; however, the NVS service itself must be protected from disruption. As mentioned above, our current user best practices of consuming NVS webservices is caching rather than continuous firing or requests. Additionally, the NVS SPARQL endpoint has a time-out for very long and complex SPARQL queries that could disrupt the service.
OTHER REQUIREMENTS	The contents of the NVS are freely available for download and local re-use.

6 SeaDataNet technical specification

6.1 Overview

6.1.1 Scope for the RI

Following the results of the questionnaires and first FAIRness analysis, the focus for SeaDataNet will be on its CDI data access service, and the SeaDataNet data product catalogue based on Sextant, plus upgrade of some elements in the exposed metadata. Similar to Euro-Argo most attention will be given to the service layers for machine to machine access, while the dataflow from the sources and the data repositories will remain out-of-scope (being internal to the SeaDataNet community). This is illustrated in the diagram below.

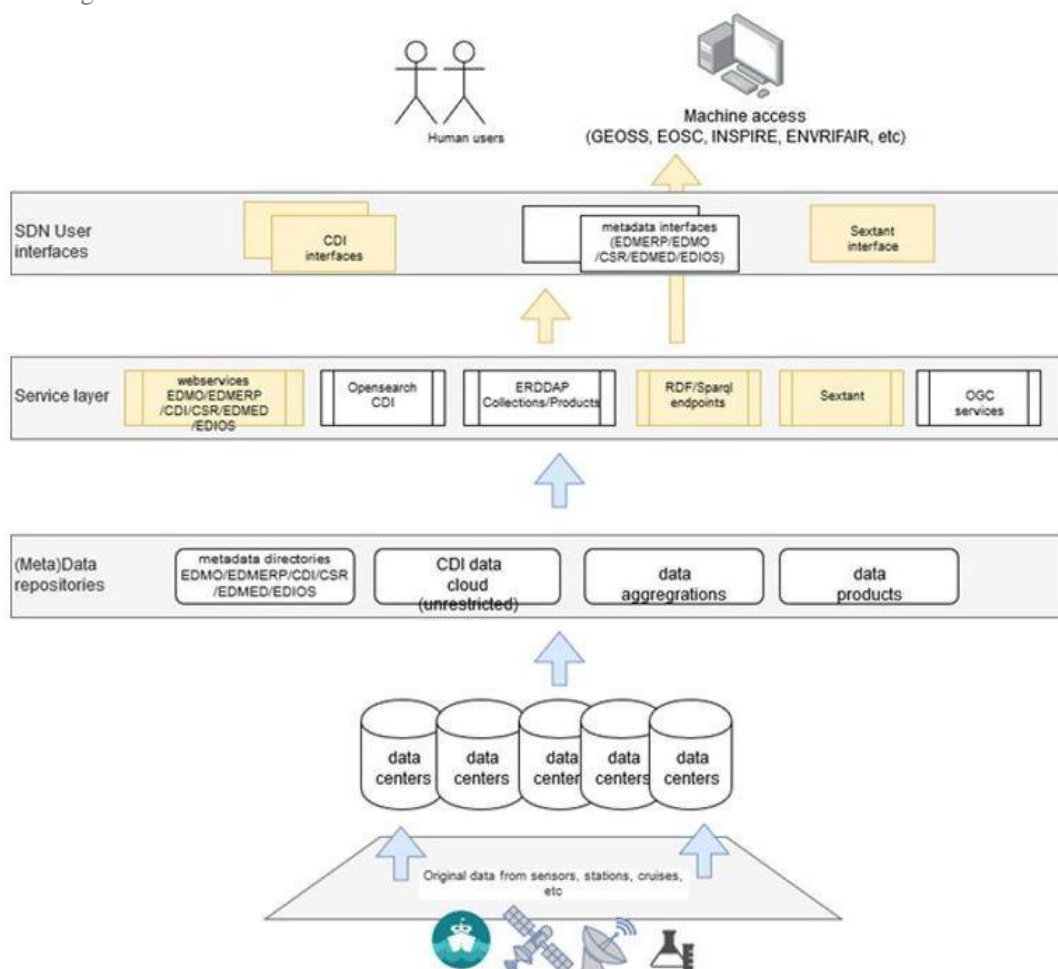


Figure 5: Schematic dataflow in SeaDataNet RI with focus areas for FAIRness

6.1.2 Summary of the technical implementation plan

As a result of the gaps identified during the FAIRness analysis the following list of implementation actions has been created as part of D9.2.

FINDABILITY	void
ACCESSIBILITY	<ul style="list-style-type: none"> • Sextant: Registration of SeaDataNet catalog in an official registry, like FAIRSharing. • CDI: API development for improved machine access. Develop a restful API to improve the ordering and download of CDI (unrestricted) data for improved machine access.

INTEROPERABILITY	<ul style="list-style-type: none"> ● Sextant: access to dataplots via WPS Develop WPS to allow data plot access, accessible from Sextant metadata <p>CDI and Sextant: Explicit persistency policy in metadata</p>
	<ul style="list-style-type: none"> ● Sextant: Upgrade metadata with vocabularies (especially concerning provenance and quality) Upgrade Sextant metadata to incorporate more vocabularies. Collect provenance information from dataproduct. After this metadata records for products and aggregations can be updated
REUSABILITY	<ul style="list-style-type: none"> ● CDI: Expand current CDI metadata Machine readable and interpretable information is needed e.g. for quality info, processing info. This info is known, but not captured and provided in structured metadata. Upgrade CDI metadata eventually to incorporate more vocabularies, regarding provenance and quality. We will use the linked data principles and SeaDataNet directories and vocabularies (linked to cruises, projects, organisations, data sets, etc) ● Sextant: Collect and expose structured provenance metadata for data products and aggregations in Sextant This information is known but not captured and provided in structured metadata yet (usually only text/pdf format). Work will be done to extract elements and store it as machine readable and interpretable information e.g. for quality info, processing steps, validation, software versions.

As indicated in the table above the developments will focus on three services: CDI metadata and API, CDI sparQL endpoint, and the Sextant data product catalogue. These services will be used in next chapters to point out the specific developments in each of them.

6.1.3 Indicative planning for delivery

Analysis of the required semantic solutions in 2020 Q2,

Implementation semantic solutions Q1 2021.

First proof of concepts in 2020 Q2,

Demo version 2021 Q1.

6.2 CDI metadata and data access API

6.2.1 Description

SERVICE PERSPECTIVE	<p>The SeaDataNet Common Data Index (CDI) is an existing service (https://cdi.seadatanet.org/) which gives users a highly detailed insight in the availability and geographical spreading of marine data sets that are managed by more than 100 marine and oceanographic data centres, located in 34 countries, around and riparian to European seas. It provides a unique interface for requesting access, and if granted, for downloading subsets from the rich and steadily increasing volume of marine data sets for physics, chemistry, geology, biology, geophysics, and bathymetry that these distributed data centres are managing. Currently, the CDI service gives access to more than 2.3 million metadata records and associated data sets, in majority in SeaDataNet standard data formats and using the SeaDataNet controlled vocabularies for many aspects, plus some elements in free text (e.g. about data quality).</p> <p>The CDI system already offers machine access to metadata (CSW/OAI-PMH), but under ENVRI-FAIR the FAIRness of the CDI service will be expanded by developing a Machine2Machine accessible API to facilitate access to data (besides current human user interface) and at the same time the metadata will be slightly expanded and upgraded to optimise machine access.</p>
FEATURES	<ul style="list-style-type: none"> • Develop an API with workflow to enable search, order and download of CDI (unrestricted/SeaDataNet licensed) data by machine users. • Draft and add to the metadata an explicit persistency policy. Upgrade the CDI metadata to incorporate more vocabularies, and linked data regarding provenance and quality information. This info is known, but not captured and provided in structured metadata and therefore not yet suitable for machine interpretation.
USER OVERVIEW	Scientific users of e.g. VRE's (programming e.g. in Jupyter notebook), as well as European/Global scale services periodically harvesting a certain subset of the CDI data resources.
TECHNOLOGIES	<p>Main technologies and standards used:</p> <ul style="list-style-type: none"> • Restful API in combination with OpenSearch • NERC Vocabulary Service • ISO19115/19139 metadata profile CDI
OPERATING ENVIRONMENT	The CDI API will be deployed on MARIS webservers, 24/7 operational with 99,9% uptime. The NVS is operated and hosted by BODC. The data behind the CDI metadata indexes is stored in the cloud at EUDAT cloud servers (B2SAFE), so there will be communication with B2SAFE (as is already operational in the human user interface)
CONSTRAINTS	Even though the published data via the API is on a light license, there is a data policy users need to accept, and an authentication is required via MARINE-ID (operated by IFREMER). This machine users of the API will need to authenticate and accept the license.
DOCUMENTATION	Metadata search via API. This will trigger an order workflow. Datasets per order will be delivered in batches, as zip file with datasets in ODV/NetCDF format (see all SDN formats on https://www.seadatanet.org/Standards/Data-Transport-Formats)
ASSUMPTIONS/DEPENDENCIES	N/A

6.2.2 New features

6.2.2.1 Feature 1 : Restful API development

DESCRIPTION AND PRIORITY	Develop a Restful API with workflow to enable search, order and download of CDI (unrestricted/SeaDataNet licensed) data by machine users.
USE CASES	<ul style="list-style-type: none"> • Programme authenticates to API via Marine-ID, receives token • Search: Submit query, receive results (list of PID's, metadata content) • Order: submit list of PID's as order, receive order-id • Check progress of order, receive status, in the end receive download URL

	<ul style="list-style-type: none"> Download: Use token and URL to download files.
ADDITIONAL REQUIREMENTS	Include Swagger documentation to support users

6.2.2.2 Feature 2 : Add persistency policy

DESCRIPTION AND PRIORITY	Draft and add to metadata an explicit machine accessible persistency policy in metadata
USE CASES	Request the persistency policy as metadata element as part of the metadata request.
ADDITIONAL REQUIREMENTS	Important to expand the CDI ISO19115 metadata model and formulate the policy.

6.2.2.3 Feature 3 : Add machine readable provenance information

DESCRIPTION AND PRIORITY	Upgrade the CDI metadata to incorporate more vocabularies regarding provenance and quality information. Machine readable and interpretable information is needed e.g. for quality info, processing info. This info is known, but not captured and provided in structured metadata.
USE CASES	<ul style="list-style-type: none"> Expand the CDI ISO19115 metadata model with elements for provenance and quality This will include SWE information links, so e.g. links to sensor descriptions created by the involved data centers. BODC to expand the NVS with vocabularies to support this information. Capture the metadata elements for quality and processing info (done by DataCenters - involvement OGS, IFREMER, CSIC, RBINS. Adding this info will be mostly forward looking, not for older records. Expose as part of metadata for demonstration cases.
ADDITIONAL REQUIREMENTS	<ul style="list-style-type: none"> Needs working group for the data model, SWE work and the NVS update (NOC) Data centers required to test: RBINS, OGS, CSIC, IFREMER

6.2.3 External interface

USER INTERFACE	There is no user interface, the API is a machine to machine service.
SOFTWARE INTERFACE	SQLServer, Restful API
HARDWARE INTERFACE	n/a
COMMUNICATION INTERFACE	The API is requested via https requests. There is authentication and authorisation via Marine-ID. The data transfer rate depends on the machine's Internet connection.

6.2.4 Nonfunctional requirements

USABILITY	The SeaDataNet ARGO monitoring will check the uptime of the service. Upon failure, an alert message is sent to the service owner.
------------------	---

OPERATIONAL	n/a
PERFORMANCE	The API is proposed for interactive services such as VRE's or processing services. The servers will expect responses to queries within one second.
SECURITY	The servers are secured and actively monitored. Backup is being created on a daily basis.
OTHER REQUIREMENTS	All services should be documented well.

6.3 CDI SPARQL endpoint

6.3.1 Description

SERVICE PERSPECTIVE	<p>Various metadata directories of SeaDataNet (Organisations EDMO, projects EDMERP,..) are already available via SPARQL endpoints. The CDI system already offers machine access to metadata (CSW/OAI-PMH), but under ENVRIFAIR in cooperation with SDC the FAIRness of the CDI service will be enhanced by developing a Machine2Machine (M2M) accessible SPARQL endpoint for the metadata. Using a linked data approach, information relevant to more than one catalog can be inherited or inferred by linkages between the catalogs (EDMO, EDMERP, CSR, NVS).</p> <p>The CDI triple store will be very large when generating this one-on-one for the 2.3M records. The challenge will therefore lie in the aggregation of the metadata, the synchronisation process to RDF.</p>
FEATURES	<ul style="list-style-type: none"> Updating the current specification of the mapping of the CDI datamodel to RDF. This specification needs to be enhanced to incorporate provenance information related to quality and processing etc. including the links to the other metadata repositories, and the NVS. Develop the SparQL endpoint for the CDI metadata
USER OVERVIEW	Scientific users, search engines, other services with links to CDI.
TECHNOLOGIES	<p>Main technologies and standards to be investigated/used:</p> <ul style="list-style-type: none"> Fuseki Java-Jena NERC Vocabulary Service RDF
OPERATING ENVIRONMENT	The CDI SPARQL endpoint will be deployed on MARIS webservers, 24/7 operational with 99,9% uptime. The data behind the CDI metadata indexes is stored in the cloud at EUDAT cloud servers (B2SAFE), so there will be communication with B2SAFE (as is already operational in the human user interface)
CONSTRAINTS	The CDI SPARQL endpoint will be fully open, no authentication.
DOCUMENTATION	CDI metadata to RDF mapping, SPARQL endpoint documentation.
ASSUMPTIONS/DEPENDENCIES	n/a

6.3.2 New features

6.3.2.1 Feature 1 : SPARQL endpoint development

DESCRIPTION AND PRIORITY	FAIRness of the CDI service will be enhanced by developing a M2M accessible SPARQL endpoint for the metadata, thereby achieving the optimised linked data result for SeaDataNet (via the links to EDMO, EDMERP, CSR, NVS, SWE). The CDI triple store will be very large when generating this one-on-one for the 2.3M records. The challenge will therefore lie in the aggregation of the metadata, and the synchronisation process to RDF.
USE CASES	<ul style="list-style-type: none"> • Develop aggregation of metadata • Map the database tables to Jena Model, using the mapping provided by Marine Institute • Export Database to multiple formats • Load RDF in Fuseki and synchronise regularly • Demonstrate the full linked data implementation with optimised data from SDN data centers CSIC, RBINS, IFREMER, OGS.
ADDITIONAL REQUIREMENTS	Include Swagger documentation for CDI sparql endpoint to follow the openAPI and support users

6.3.2.2 Feature 2 : Aggregation and mapping to RDF

DESCRIPTION AND PRIORITY	Create the mapping to RDF for CDI, including the links to the other metadata repositories, the NVS and provenance information (see other action)
USE CASES	<ul style="list-style-type: none"> • Create a useful aggregation of the CDI metadata (e.g. per parameter group, per sea-area) • RDF mapping for CDI already developed. Needs to be expanded for quality and provenance information. Plus create the dynamic mapping and use this in the SPARQL endpoint (see feature 1)
ADDITIONAL REQUIREMENTS	n/a

6.3.3 External interface

USER INTERFACE	Apache Jena Fuseki is a SPARQL server and has a user interface for server monitoring and administration.
SOFTWARE INTERFACE	SPARQL endpoint (Fuseki)
HARDWARE INTERFACE	..
COMMUNICATION INTERFACE	The SPARQL endpoint is requested via https requests. There is no authentication and authorisation required. The data transfer rate depends on the machines Internet connection.

6.3.4 Nonfunctional requirements

USABILITY	The SeaDataNet ARGO monitoring will check the uptime of the service. Upon failure, an alert message is sent to the service owner.
OPERATIONAL	..
PERFORMANCE	The SPARQL endpoint is proposed for search engines, and scientific users making complex queries. The users will expect answers to queries within a few seconds.

SECURITY	Caching of query results locally is the preferred way to work with a sparql endpoint to avoid performance issues.
	The servers are secured and actively monitored. Backup is being created on a daily basis. The sparql endpoint is a read-only service
OTHER REQUIREMENTS	All services should be documented well. Tests will be done together with the data centers (CSIC, RBINS, OGS, IFREMER) if the provenance and quality information is indeed exposed and flowing to the users.

6.4 Sextant data product catalogue upgrading

6.4.1 Description

SERVICE PERSPECTIVE	<p>SeaDataNet provides aggregated datasets (ODV collections of all unrestricted SeaDataNet measurements of temperature and salinity by sea basins) and climatologies (regional gridded field products) based on the aggregated datasets and data from external data sources such as the COriolis Ocean Dataset for Reanalysis (CORA) and the World Ocean Database (WOD) for all the European sea basins and the Global Ocean. Each SeaDataCloud product is described in a Product Information Document (PIDoc) that can be accessed from the product's landing page.</p> <p>The data product metadata is accessible and searchable via the Sextant catalogue that already exists (https://www.seadatanet.org/Products#/search?from=1&to=30). As part of ENVRI-FAIR the FAIRness of the catalogue has been analysed, leading to a set of required upgrades that are specified here.</p>
FEATURES	<ul style="list-style-type: none"> • Registration of SeaDataNet catalogue in an official registry. • Add an explicit persistency policy in metadata. • Create access to dataplots via WPS • Develop WPS to allow data plot access, accessible from Sextant metadata • Upgrade metadata with vocabularies (especially concerning provenance and quality). Collect provenance information from dataproduct. After this metadata records for products and aggregations can be updated. • Collect and expose structured provenance metadata for data products and aggregations in Sextant. This information is known but not captured and provided in structured metadata yet (usually only text/pdf format). Work will be done to extract elements and store it as machine readable and interpretable information e.g. for quality info, processing steps, validation, software versions.
USER OVERVIEW	Scientific users, search engines, other services accessing the aggregations and dataproduct (e.g. for cloud processing/VRE).
TECHNOLOGIES	<p>Main technologies used:</p> <ul style="list-style-type: none"> • Sextant (based on Geonetwork) • NERC Vocabulary Server • WPS/ERDDAP
OPERATING ENVIRONMENT	The Sextant service is already operational on the IFREMER webservers.
CONSTRAINTS	Sextant metadata is fully open, no authentication. Access to the datafiles required authentication via MARINE-ID.
DOCUMENTATION	Sextant metadata format.
ASSUMPTIONS/DEPENDENCIES	-

6.4.2 New features

6.4.2.1 Feature 1 : Registration of SeaDataNet catalogue in an official registry

DESCRIPTION AND PRIORITY	To increase the visibility of the catalogue it is important to register it in an official registry.
USE CASES	<ul style="list-style-type: none"> • Explore best suitable registries (e.g. FAIRsharing, B2Find, ...) • Register.
ADDITIONAL REQUIREMENTS	none.

6.4.2.2 Feature 2 : Add an explicit persistency policy in metadata

DESCRIPTION AND PRIORITY	Draft and add to metadata an explicit machine accessible persistency policy in metadata
USE CASES	<ul style="list-style-type: none"> • Test by requesting the persistency policy as element as part of the metadata request
ADDITIONAL REQUIREMENTS	Important to expand the Sextant ISO19115 metadata model and formulate the policy.

6.4.2.3 Feature 3 : Create access to dataplots via WPS

DESCRIPTION AND PRIORITY	Develop a WPS to allow data plot access, accessible from Sextant metadata
USE CASES	<ul style="list-style-type: none"> • Discover metadata • From link in metadata access the data access service • Request subset
ADDITIONAL REQUIREMENTS	n/a

6.4.2.4 Feature 4 : Upgrade metadata elements with vocabularies

DESCRIPTION AND PRIORITY	Upgrade metadata elements with vocabularies especially concerning provenance and quality. Collect provenance information from data product generation process. After this, the metadata records for products and aggregations can be updated.
USE CASES	<ul style="list-style-type: none"> • Expand metadata format elements • Identify and create new NVS vocabularies • Publish in metadata
ADDITIONAL REQUIREMENTS	Important to expand the Sextant ISO19115 metadata model and formulate the policy.

6.4.2.5 Feature 5 : collect provenance information (QC and processing)

DESCRIPTION AND PRIORITY	Collect and expose structured provenance metadata for data products and aggregations in Sextant. This information is known but not captured and provided in structured metadata yet (usually only text/pdf format). Work will be done to extract elements and store it as machine readable and interpretable information e.g. for quality info, processing steps, validation, software versions.
---------------------------------	--

USE CASES	<ul style="list-style-type: none"> • Set up working group with the WebODV and DIVA developers • Collect the required elements during processing (in WebODV and DIVA) • Publish in metadata • https://www.w3.org/TR/vocab-dqv/#dqv:QualityMetadata is a candidate quality metadata ontology
ADDITIONAL REQUIREMENTS	-

6.4.3 External interface

USER INTERFACE	https://www.seadatanet.org/Products#/search?from=1&to=30
SOFTWARE INTERFACE	Sextant (Geonetwork), WPS (ERDDAP?)
HARDWARE INTERFACE	..
COMMUNICATION INTERFACE	Sextant is requested via https requests. There is no authentication and authorisation required for the metadata discovery. For data access MARINE-ID authentication is required.

6.4.4 Nonfunctional requirements

USABILITY	The SeaDataNet ARGO monitoring will check the uptime of the service. Upon failure, an alert message is sent to the service owner.
OPERATIONAL	..
PERFORMANCE	Sextant is proposed for scientific users discovering data collections and quality checked data products. The users will expect answers to queries within a few seconds.
SECURITY	The servers are secured and actively monitored. Backup is being created on a daily basis.
OTHER REQUIREMENTS	All services should be documented well.

7 Conclusion

The developments specified in this document D9.3 will significantly improve the Machine to Machine fairness of Marine domain RIs data system. The task 9.8 « Demonstrate Marine subdomain FAIRness, EOv global product » is based on the APIs developed in ENVRI-FAIR and specified in chapters 2 to 6 of this document.

As a user of Marine domain services, the D9.8 demonstrator will query, subset and aggregate data files from the 5 RIs on specific EOVs (Essential Ocean Variables).

The D9.8 demonstrator will illustrate the fairness of Marine domain Machine to Machine data and metadata access.

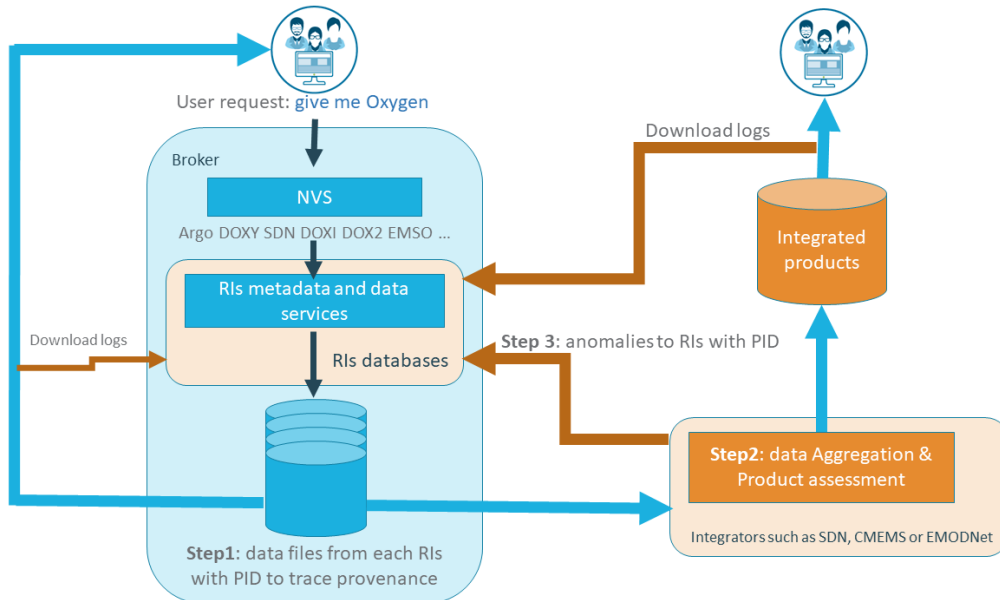


Figure 8: task 9.8 demonstrator, a user requests data files containing a parameter (such as oxygen) from the 5 marine domain Research Infrastructures

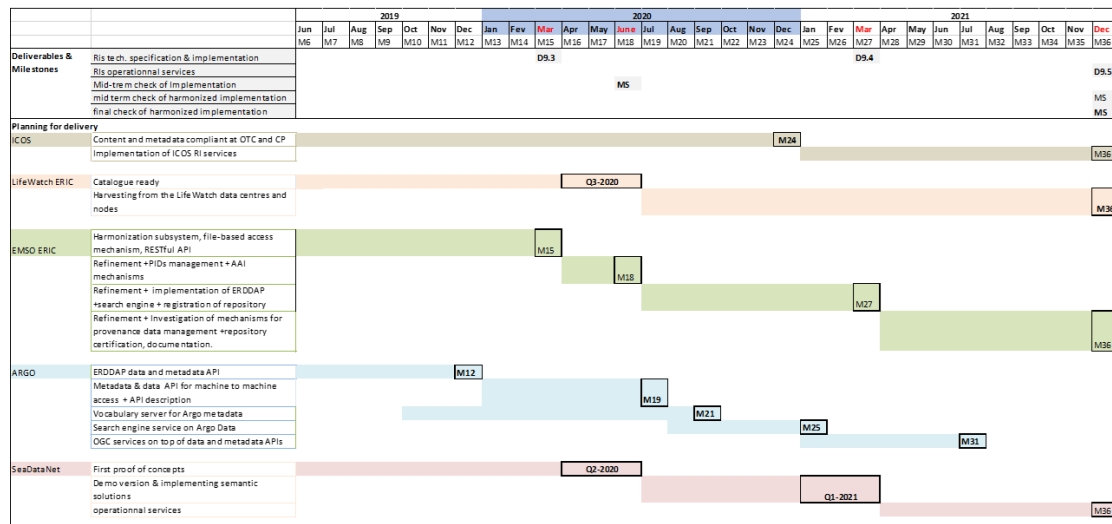


Figure 9: Gantt diagram for RIs technical implementation

8 References

Ref	Title	Version / Date
D9.1	Marine subdomain FAIRness roadmap https://iagos-comm.iek.fz-juelich.de/dmsf/files/3946/view	V2.0 August 2019
D9.2	Marine subdomain implementation plan https://iagos-comm.iek.fz-juelich.de/dmsf/files/3944/view	V1.0 November 2019

9 Appendices

9.1 Appendix 1: Glossary

API	Application Programming Interface
B2HANDLE	EUDAT minting, storing, managing and accessing persistent identifiers
CAS	Central Authentication Service
CDI	Common Data Index (metadata format and data access system by SeaDataNet)
CF	Climate and Forecast (semantics for NetCDF)
CMEMS	Copernicus Marine Environment Monitoring Service
COPERNICUS	A major earth observation programme run by European Commission and European Space Agency
CP	Carbon Portal
CSR	Cruise Summary Report
CSW	Catalogue Service for the Web
DMP	1) Data Management Plan 2) Data Management Platform (WP9)
DOI	Digital Object Identifier
DSA	Data Seal of Approval
ECV	Essential Climate Variable
EDIOS	European Directory of ocean Observing Systems
EDMED	European Directory of Marine Environmental Datasets (SeaDataNet)
EDMO	European Directory of Marine Organisations
EDMERP	European Directory of Marine Environmental Research Projects
EMODNET	European Marine Observation and Data Network
EMSO	European Multidisciplinary Seafloor and water column Observatory
ENVRI	1) An environmental RI cluster FP7 project 2) Environment research infrastructures (in ESFRI level or upcoming) as a community
ENVRIplus	An environmental RI cluster H2020 project
EOSC	European Open Science Cloud
EOV	Essential Ocean Variable(s)
ERDDAP	NOAA developed science data server technology
ERIC	European Research Infrastructure Consortium (legal entity type)
ESFRI	European Strategy Forum on Research Infrastructures

FAIR	Findable Accessible Interoperable Reusable
GBIF	Global Biodiversity Information Facility
GCMD	Global Change Master Directory
GDAC	Global Data Assembly Center
GEMET	GEneral Multilingual Environmental Thesaurus
GEO	Group on Earth Observation (System of Systems)
GEOSS	Global Earth Observation System of Systems
GOFAIR	An international programme on FAIR implementation
GUI	Graphical User Interface
ICOS	Integrated Carbon Observation System
ICT	Information and Communications Technology
IMIS	Integrated Marine Information System
INSPIRE	Infrastructure for Spatial Information in the European Community
iRODS	Open Source Data Management Software
JCOMM	Joint Technical Commission for Oceanography and Marine Meteorology
M	Month
Marine-ID	Registration and authentication services for marine data services
MDA	Marine Data Archive
NetAPP	Hybrid cloud service
NetCDF	Network Common Data Format
NVS	NERC Vocabulary Services
NOAA	US National Oceanic and Atmospheric Administration
OAUTH	Open Authorization (standard)
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
OBIS	Ocean Biogeographic Information System
OGC	Open Geospatial Consortium
OpenDAP	Open-source Project for a Network Data Access Protocol
ORCID	Open Researcher and Contributor ID
OWL	Web Ontology Language
PID	Persistent Identifiers

POC	Proof of Concept
PROV-O	Web Ontology Language encoding of the PROV Data Mode
Q	Quarter
QA/QC	Quality Assurance/Quality Control
RDF	Resource Description Framework
RI	Research Infrastructure
RSS	Really Simple Syndication
SAML	Security Assertion Markup Language
SEADATANET	SeaDataNet pan-European infrastructure for marine data management
SME	Small or medium Enterprise
SparQL	SparQL Protocol and RDF Query Language
SWOT	Analysis on Strengths, Weaknesses, Opportunities and Threats
VRE	Virtual Research Environment
WoRMS	World Registry of Marine Species
WPS	Web Processing Services

9.2 Appendix 2: Euro-Argo APIs examples

Example of OpenSearch Atom output

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/"
  xmlns:georss="http://www.georss.org/georss">
  <title>Argo floats search</title>
  <link href="https://argofloatsApi.eu.org/api/floats/search"/>
  <updated>2020-03-05T14:30:02Z</updated>
  <author>
    <name>Argo GDAC</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
  <opensearch:totalResults>1</opensearch:totalResults>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>1</opensearch:itemsPerPage>
  <opensearch:Query
    role="request"
    searchTerms="6902964"
    startPage="1"
    geo:box="-74.0667,40.69418,-73.9116,40.7722"/>
  <link rel="alternate" href="https://argofloatsApi.eu.org/api/floats/search?q=6902964&box=-74.0667,40.69418,-73.9116,40.7722" type="text/html"/>
  <link rel="search" type="application/opensearchdescription+xml"
    href="http://example.com/opensearchdescription.xml"/>
  <georss:box>40.69418 -74.0667 40.7722 -73.9116</georss:box>
  <entry>
    <title>Float 6902964</title>
    <link href="https://fleetmonitoring.euro-argo.eu/api/float/6902964"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <dc:identifier>1225c695-cfb8-4ebb-aaaa-80da344efa6a</dc:identifier>
    <updated>2019-12-13T18:30:02Z</updated>
    <georss:line>40.73763 -73.9972 40.73519 -73.99167 40.737015 - 73.99035 40.73643 -73.98914 40.734640 -73.990431 40.731617 -
73.991504</georss:line>
  </entry>
</feed>
```

Example of OpenSearch “profiles/search” Atom output

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/"
  xmlns:georss="http://www.georss.org/georss">
  <title>Argo floats search</title>
  <link href="https://argofloatsApi.eu.org/api/profiles/search"/>
  <updated>2020-03-05T14:30:02Z</updated>
  <author>
    <name>Argo GDAC</name>
  </author>
  <id>urn:uuid:60a76c80-d399-11d9-b93C-0003939e0af6</id>
  <opensearch:totalResults>1</opensearch:totalResults>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>1</opensearch:itemsPerPage>
  <opensearch:Query
    role="request"
    searchTerms="6902964"
    startPage="1"
    geo:box="-74.0667,40.69418,-73.9116,40.7722"/>
  <link rel="alternate" href="https://argofloatsApi.eu.org/api/profiles/search?q=6902964&box=-74.0667,40.69418,-73.9116,40.7722" type="text/html"/>
  <link rel="search" type="application/opensearchdescription+xml"
    href="http://example.com/opensearchdescription.xml"/>
  <georss:box>40.69418 -74.0667 40.7722 -73.9116</georss:box>
  <entry>
    <title>Float 6902964 profile #1</title>
    <link href="https://fleetmonitoring.euro-argo.eu/float/6902964/profile/1"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <dc:identifier>1225c695-cfb8-4ebb-aaaa-80da344efa6a</dc:identifier>
    <updated>2019-12-13T18:30:02Z</updated>
    <georss:line>40.73763 -73.9972 40.73519 -73.99167 40.737015 - 73.99035 40.73643 -73.98914 40.734640 -73.990431 40.731617 -
73.991504</georss:line>
  </entry>
</feed>
```

Example of Argo API “get profiles” output

```
{
  "errorCode":0,
  "errorMessage":"","
  "query":{
    "downsampling":"none",
    "samplingparameter":"none",
    "platformCode":"2901853",
    "parameterCode":"70",
    "measureType":"1",
    "startTimeMillis":"1394650350000",
    "endTimeMillis":"1394650350000",
    "startTimeFormat":"2014-03-12T18:52:30Z",
    "endTimeFormat":"2014-03-12T18:52:30Z"
  },
  "result":[
```

```

{
  "observationId": "32596535",
  "dataSetSizeBeforeRegression": 990,
  "dataSetSizeAfterRegression": 990,
  "dateTime": "1394650350000",
  "dataType": "PF",
  "dataSetSize": 990,
  "latitude": 6.48,
  "longitude": 62.391,
  "dateQc": 1,
  "positionQc": 1,
  "zCode": 28,
  "data": [
    [
      35.19198,
      4.1,
      1,
      1,
      1
    ],
    [
      35.19193,
      6,
      1,
      2,
      1
    ],
    ...
  ]
}

```

Example of Argo API “get trajectory” output in Annex 9.2

```

{
  "errorCode": 0,
  "errorMessage": "",
  "query": {
    "downsampling": "LTTB",
    "samplingparameter": "400",
    "platformCode": "6200450",
    "parameterCode": "35",
    "measureType": "15",
    "startTimeMillis": "1569110400000",
    "endTimeMillis": "1571702400000",
    "startTimeFormat": "2019-09-22T00:00:00Z",
    "endTimeFormat": "2019-10-22T00:00:00Z"
  },
  "result": [
    {
      "name": "6200450-0",
      "dataSetSizeBeforeRegression": 2109,
      "dataSetSizeAfterRegression": 400,
      "data": [
        [
          1569110717000,
          16.24, 45.24, -3.50,
          1
        ],
        [
          1569113121000,
          16.22, 45.24, -3.51,
          1
        ],
        ...
      ],
      "z_value": "3.0"
    }
  ]
}

```

Example of Argo API “get timeseries” output in annex 9.2

```

{
  "errorCode": 0,
  "errorMessage": "",
  "query": {
    "downsampling": "LTTB",
    "samplingparameter": "400",
    "platformCode": "6200450",
    "parameterCode": "35",
    "measureType": "15",
    "startTimeMillis": "1569110400000",
    "endTimeMillis": "1571702400000",
    "startTimeFormat": "2019-09-22T00:00:00Z",
    "endTimeFormat": "2019-10-22T00:00:00Z"
  },
  "result": [
    {
      "name": "6200450-0",
      "dataSetSizeBeforeRegression": 2109,
      "dataSetSizeAfterRegression": 400,
      "data": [
        1569110717000,

```

```

        16.24,
        1
      ],
      [
        1569113121000,
        16.22,
        1
      ],
      ...
    ],
    "z_value": "3.0"
  }
}
}

```

Example of Argo “get profiles” output

```

{
  "errorCode": 0,
  "errorMessage": "",
  "query": {
    "downsampling": "none",
    "samplingparameter": "none",
    "platformCode": "2901853",
    "parameterCode": "70",
    "measureType": "1",
    "startTimeMillis": "1394650350000",
    "endTimeMillis": "1394650350000",
    "startTimeFormat": "2014-03-12T18:52:30Z",
    "endTimeFormat": "2014-03-12T18:52:30Z"
  },
  "result": [
    {
      "observationId": "32596535",
      "dataSetSizeBeforeRegression": 990,
      "dataSetSizeAfterRegression": 990,
      "date": "1394650350000",
      "dataType": "PF",
      "dataSetSize": 990,
      "latitude": 6.48,
      "longitude": 62.391,
      "dateQc": 1,
      "positionQc": 1,
      "zCode": 28,
      "data": [
        [
          35.19198,
          4.1,
          1,
          1,
          1
        ],
        [
          35.19193,
          6,
          1,
          2,
          1
        ],
        ...
      ]
    }
  ]
}

```

Example of Argo “get trajectories” output

```

{
  "errorCode": 0,
  "errorMessage": "",
  "query": {
    "downsampling": "LTTB",
    "samplingparameter": "400",
    "platformCode": "6200450",
    "parameterCode": "35",
    "measureType": "15",
    "startTimeMillis": "1569110400000",
    "endTimeMillis": "1571702400000",
    "startTimeFormat": "2019-09-22T00:00:00Z",
    "endTimeFormat": "2019-10-22T00:00:00Z"
  },
  "result": [
    {
      "name": "6200450-0",
      "dataSetSizeBeforeRegression": 2109,
      "dataSetSizeAfterRegression": 400,
      "data": [
        [
          1569110717000,
          16.24, 45.24, -3.50,
          1
        ],
        [
          1569113121000,
          16.22, 45.24, -3.51,
          1
        ]
      ]
    }
  ]
}

```



```

    ],
    ...
  },
  "z_value": "3.0"
}
}
}

```

Example of Argo “get timeseries” output

```

{
  "errorCode": 0,
  "errorMessage": "",
  "query": {
    "downsampling": "LTTB",
    "samplingparameter": "400",
    "platformCode": "6200450",
    "parameterCode": "35",
    "measureType": "15",
    "startTimeMillis": "1569110400000",
    "endTimeMillis": "1571702400000",
    "startTimeFormat": "2019-09-22T00:00:00Z",
    "endTimeFormat": "2019-10-22T00:00:00Z"
  },
  "result": [
    {
      "name": "6200450-0",
      "dataSetSizeBeforeRegression": 2109,
      "dataSetSizeAfterRegression": 400,
      "data": [
        [
          1569110717000,
          16.24,
          1
        ],
        [
          1569113121000,
          16.22,
          1
        ],
        ...
      ],
      "z_value": "3.0"
    }
  ]
}

```

Example of OGC-SensorThing Thing feature

```

{
  "@iot.id": 1,
  "@iot.selfLink":
  "http://example.org/v1.0/Things(1)",
  "Locations@iot.navigationLink":
  "Things(1)/Locations",
  "Datastreams@iot.navigationLink":
  "Things(1)/Datastreams",
  "HistoricalLocations@iot.navigationLink":
  "Things(1)/HistoricalLocations",
  "name": "Arvor 690078",
  "description": "This
  thing is an Argo float.",
  "properties": {
    "institution": "Ifremer",
    "floatType": "NKE Arvor"
  }
}

```

Example of a OGC-sensorThing location entity

```

{
  "@iot.id": 1,
  "@iot.selfLink":
  "http://example.org/v1.0/Locations(1)",
  "Things@iot.navigationLink":
  "Locations(1)/Things",
  "HistoricalLocations@iot.navigationLink":
  "Locations(1)/HistoricalLocations",
  "encodingType":
  "application/vnd.geo+json",
  "name": " Arvor 690078",
  "description": "Argo profiling float profile n°102",
  "location": {
    "type":
    "Feature",

```

```

    "geometry":{
      "type": "Point",
      "coordinates": [-114.06,51.05]
    }
  }
}

```

Example of a OGC-sensorThing HistoricalLocation

```

{
  "value": [
    {
      "@iot.id": 1,
      "@iot.selfLink":
      "http://example.org/v1.0/HistoricalLocations(1)",
      "Locations@iot.navigationLink":
      "HistoricalLocations(1)/Locations",
      "Thing@iot.navigationLink": "HistoricalLocations(1)/Thing",
      "time": "2015-01-25T12:00:00-07:00"
    },
    {
      "@iot.id": 2,
      "@iot.selfLink":
      "http://example.org/v1.0/HistoricalLocations(2)",
      "Locations@iot.navigationLink":
      "HistoricalLocations(2)/Locations",
      "Thing@iot.navigationLink":
      "HistoricalLocations(2)/Thing",
      "time": "2015-01-25T13:00:00-07:00"
    }
  ],
  "@iot.nextLink": "http://example.org/v1.0/Things(1)/HistoricalLocations?$skip=2&$stop=2"
}

```

Example of OGC-sensorThing data stream

```

{
  "@iot.id": 1,
  "@iot.selfLink":
  "http://example.org/v1.0/Datastreams(1)",
  "Thing@iot.navigationLink":
  "HistoricalLocations(1)/Thing",
  "Sensor@iot.navigationLink":
  "Datastreams(1)/Sensor",
  "ObservedProperty@iot.navigationLink":
  "Datastreams(1)/ObservedProperty",
  "Observations@iot.navigationLink":
  "Datastreams(1)/Observations",
  "name": "Sea water temperature from Argo float",
  "description": "This is a datastream measuring the sea water temperature from an Argo float.",
  "unitOfMeasurement":
  {
    "name":
    "degree Celsius",
    "symbol":
    "°C",
    "definition":
    "http://units-of-measure.org/ucum.html#para-30"
  },
  "observationType":
  "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
  "observedArea": {
    "type":
    "Polygon",
    "coordinates":
    [[[100,0],[101,0],[101,1],[100,1],[100,0]]]
  },
  "phenomenonTime":
  "2014-03-01T13:00:00Z/2015-05-11T15:30:00Z",
  "resultTime":
  "2014-03-01T13:00:00Z/2015-05-11T15:30:00Z"
}

```