D10.4 EOSC Hub Technical Architecture and standards roadmap v2

Lead Partner:	INFN
Version:	2
Status:	Under EC review
Dissemination Level:	Public
Document Link:	https://documents.egi.eu/document/3495

Deliverable Abstract

This document describes the EOSC-hub contribution to the definition of the EOSC Technical Architecture, which is currently being developed by the EOSC architecture Working Group. It is based on the concepts of service interoperability and end-to-end composition of services and foresees the definition of a reference architecture in which EOSC building blocks and the main functions, interfaces, APIs and standards are identified. This architecture is expected to facilitate access to services, lower the barriers to integrate and composes services and promote the usage of services between adjacent communities.

As a basis for the proposed architecture, service categories have been introduced, mapping their functions, relationships and organisation to the kinds of services required for the federating core of EOSC and the external EOSC service portfolio. The concept of the end-to-end composition of services has been presented, highlighting

EOSC-HUB RECEIVES FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME UNDER GRANT AGREEMENT NO. 777536.

the most common integration scenarios. Leveraging the defined service categories and on the concepts of service interoperability and composition, a reference EOSC Technical Architecture has been defined identifying a hierarchical structure where the first level relies on service categories (Federation & Access enabling,, Common and Thematic), the second level introduces functional categories, that groups technical functions to facilitate their identification, and the third is made of the technical functions that has been called building blocks.

EOSC-hub is working on defining the building blocks of the architecture for each service type and specified a common approach to complete this task. It foresees the identification of the main building blocks/technical functions in each service category and, for each of those, the definition of a technical specification that includes a high-level architecture, suggested EOSC standards and APIs and interoperability guidelines. As a consequence, interoperability between services compliant with the EOSC specifications will be easier to be achieved.

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	Giacinto Donvito, Diego Scardaci and Mark Van De Sanden	INFN/WP10, EGI.eu/WP10 & SurfSARA/WP10	01/08/2019
Moderated by:	Malgorzata Krakowian	EGI.eu/WP1	
Reviewed by	Daan Broeder Baptiste Grenier	KNAW EGI.eu	15/09/2019
Approved by:	AMB		

DOCUMENT LOG

Issue	*Date*	*Comment*	*Author*
v.0.1	01/08/2019	Full draft	Diego Scardaci (EGI.eu), Giacinto Donvito (INFN) and Mark Van De Sanden (SurfSARA) Lukas Dutka (Cyfronet) Giuseppe Fiameni (CINECA) Heinrich Widmann (DRKZ) Ignacio Blanquer (UPV) Enol Fernandez (EGU.eu) Joao Pina (LIP) Marica Antonacci (INFN) Marcin Plociennik (PSNC) Jens Jensen (STFC) Michal Prochazka (CESNET) Licia Florio (GEANT)
v. 0.2	20/08/2019	New version answering to reviewers' comments	Diego Scardaci (EGI.eu), Giacinto Donvito (INFN) and Mark Van De Sanden (SurfSARA)
v. 0.3	27/08/19	Proofing Section 5 abstract exec summary, intro.	Owen Appleton (EGI.eu)
v. 0.4	13/12/2019	New version answering to EC comments	Stefano Nicotri (INFN), Alessandro Costantini (INFN), Giacinto Donvito (INFN)

TERMINOLOGY

https://wiki.eosc-hub.eu/display/EOSC/EOSC-hub+Glossary

Terminology/Acronym	*Definition*
Access Enabling services	Delivering features allowing customers to easily exploit EOSC resources such as discovery, ordering and workflow enabling services (e.g. the EOSC Portal).
AAI	Authentication and Authorisation infrastructure
Building block	Technical functions that can be offered by one or more services. A building block is defined through a technical specification that includes an high-level architecture, suggested EOSC standards and APIs and interoperability guidelines
Common services	Providing generic capabilities usable by any science discipline each supporting aspects of the data lifecycle from creation to processing, analysis, preservation, access and reuse. Examples of services belonging to this category are multi-disciplinary services for data discovery, processing, workflow management and orchestration, data management, etc.
CMDB	Configuration Management Database
Federation services	Needed to operate the EOSC (e.g. a common helpdesk, accounting information gathering, monitoring)
HPC	High Parallel Computing
HTC	High Throughput Computing
IaaS	Infrastructure as a Service
Interoperability	Ability of two or more services to work together to deliver a feature for users.
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
PaaS	Platform as a Service
Reference architecture	In the field of software architecture or enterprise architecture, reference architecture provides a template solution for architecture for a particular domain. It provides a common vocabulary with which to discuss implementations, often with the aim of stressing commonalities.
Service composability	Ability to compose services to create new workflow.

Thematic services	Community-specific capabilities including research core data, data products, scientific software, and pipelines. Examples of thematic services are data resources and software tools to access study and compare the data; data brokering services tailored to the needs of specific scientific communities.
UR	Accounting Usage Record
VM	Virtual Machine

Contents

1 Introduction 11
2 Landscape 13
2.1 EC Implementation Roadmap 13
2.2 EOSC Architecture Working Group 14
2.3 Past work on the EOSC Technical Architecture 15
2.3.1 EOSC Pilot Service Architecture 15
2.3.2 EOSC-hub Technical Architecture v1 16
3 The EOSC Portfolios and the EOSC Federating Core 18
4 Service Composability 20
4.1 Fostering the service interoperability 20
4.2 Federating thematic services in the EOSC 22
5 Defining the EOSC Technical Architecture 23
5.1 Reference Architecture 23
5.1.1 Approach to define building blocks 24
5.1.2 Technical Specification template 26
5.2 Proposed EOSC Technical Architecture 26
5.2.1 EOSC Access Enabling and Federation services 27
5.2.2 EOSC Common services 32
5.2.3 EOSC Thematic services 35
5.2.4 EOSC Portal 36
5.2.5 Architecture governance 38
6 Use cases to drive the identification and the specification of the building blocks 39
7 Relationship with the EOSC Architecture Working Group 42
8 Technical Specifications for Federation services 43
8.1 AAI 43
8.1.1 AAI 43
8.2 Federation Tools 48
8.2.1 Helpdesk 48
8.2.2 Accounting 53
8.2.3 Monitoring 58
8.2.4 Software Quality Assurance 67

8.3 Security 71
8.3.1 Security Incident Response Trust Framework for Federated Identity (SIRTFI) 71
9 Technical Specifications for Common services 75
9.1 Data Publishing and Open Data 75
9.1.1 Digital Repository 75
9.2 Metadata Management and Data Discovery 84
9.2.1 Metadata Cataloguing and Management 84
9.2.2 Data Discovery and Access 87
9.3 Cloud Compute, Containers and Orchestration 90
9.3.1 Cloud IaaS VM Management 90
9.3.2 Cloud IaaS Container Management 92
9.3.3 Cloud IaaS Orchestration 94
9.4 PaaS solutions 96
9.4.1 PaaS Orchestration 96
9.5 Workflow Management and User Interfaces and Data Analytics 100
9.5.1 Marketplace 100
10 Conclusions and next steps 103

Executive summary

This document describes the EOSC-hub contribution to the definition of the EOSC Technical Architecture which is currently being developed by the EOSC architecture Working Group. It is based on the concepts of service interoperability and end-to-end composition of services and foresees the definition of a reference architecture where all the EOSC main functions, interfaces, APIs and standards are identified. This reference architecture will increase the added value provided by EOSC and foster its uptake, facilitating access to services, lowering barriers to integrate and composes services and promoting the usage of services between adjacent communities.

This work has taken into account the surrounding landscape and has followed the recommendations on the EOSC architecture of the European Commission described in the Staff Working Document on the *Implementation roadmap for the European Open Science Cloud* (EOSC)[1] and the mandate of the EOSC Architecture Working Group (WG)[2], recently launched by the EOSC governance. This paved the way for a refinement and better focusing of the scope of the EOSC architecture work within EOSC-hub. EOSC-hub is member of the EOSC Architecture Working Group and will contribute to the discussion within this WG. This deliverable must be seen in the light of the EOSC-hub contribution to the EOSC Architecture WG. Past work on this topic has been also analysed, notably the EOSC Service Architecture proposed by the EOSCpilot project[3].

The proposed architecture is organised according to service categories: Federation & Access enabling, Common and Thematic services. As a basis to describe the architecture, service categories have been introduced, mapping their functions, relationships and organisation to the kinds of services required for the federating core of EOSC and the external EOSC service portfolio. The concept of end-to-end service composition has been presented, highlighting the most common integration scenarios and how services belonging to different categories can cooperate to create added-value solutions for research. EOSC-hub effort to foster service interoperability and the impact of the service composability on federating thematic services into the EOSC has also been depicted.

Leveraging the service categories and on the concepts of service interoperability and composition, a proposal for a reference Technical Architecture for EOSC has been defined identifying a hierarchical structure. The first level of this hierarchy relies on the subdivision in categories and allows to differentiate services according to their function within EOSC: Federation and Access enabling are key services to operate the EOSC (e.g. the EOSC Portal or the accounting infrastructure), Common services offer add-value features on top of EOSC resources (computing, storage, data, etc) and can be reused by a multitude of other services, Thematic services implement discipline specific features and are provided directly by scientific communities. The second level of the hierarchy introduces the functional categories that groups technical functions to facilitate their identification (e.g. Authentication and Authorisation or Monitoring for federation services, Cloud

[1] Commission Staff Working Document - Implementation roadmap for the European Open Science Cloud: https://ec.europa.eu/research/openscience/pdf/swd_2018_83_f1_staff_working_paper_en.pdf

[2] https://www.eoscsecretariat.eu/working-groups/architecture-working-group

[3] https://www.eoscpilot.eu/

Compute and Metadata management for common services). In the case of thematic services, the functional categories are identified per scientific discipline. The third level is made of the technical functions that have been called building blocks. Examples of building blocks are AAI and accounting infrastructure for federation services, Cloud Infrastructure as a Service (IaaS) Virtual Machine (VM) management, a Platform as a Service (PaaS) solution or a Data Repository for common services, scientific workflows for thematic services.

EOSC-hub is working on defining the building blocks of the architecture for each service type and specified a common approach to complete this task. It foresees the identification of the main **building blocks/technical functions** in each service category. As described above, the typology of the building blocks changes according to the category they belong to. Then, for each of those building blocks, a **technical specification** that includes a **high-level architecture, suggested EOSC standards and APIs and interoperability guidelines** will be defined. As a consequence, thanks to the provided guidelines, interoperability between services offering the same technical function(s) and following the EOSC specifications will be easier to achieve, examples are the Authentication and Authorisation Infrastructure (AAI) services compliant with the AARC blueprint architecture and guidelines or monitoring and/or accounting systems able to exchange/share information and provide integrated views to the EOSC customers and service providers. This approach is tailored to the varied environment seen in EOSC, where many solutions to satisfy a given technical requirement already exist. Furthermore, having well defined EOSC endorsed standards and APIs and related interoperability guidelines for each of the identified building blocks will foster the end-to-end composition of services, lowering the barriers to make services interoperable. Indeed, other building blocks/services offering different technical functions can interoperate thanks to the EOSC interfaces, described in the technical specification (e.g. it would be easier for a thematic service integrating a common services if clear interoperability guidelines are available).

The EOSC interoperability guidelines that are being defined in the context of this work will take into account and will be based on existing community practices, well-known standards and interfaces. They should be defined by all relevant EOSC stakeholders (communities, e-infrastructures, etc.) in a collaborative manner and their adoption should not be mandatory but a natural consequence of the advantages, for a service, generated by being compliant.

In the proposed architecture, identifying building blocks and the related technical specifications for all the service categories has proved to be complex and long work; therefore we decided to follow an iterative approach starting from the functions that are more requested by the EOSC use cases[4]. Also the technical specifications, initially prepared by the technical experts within the EOSC-hub project, will be iteratively improved collecting feedback from external people with expertise in the area and involving them in the maintenance and evolution of such specifications.

[4] EOSC-hub is taking into account in this work requirements collected from EOSC Pilot Scientific Demonstrator (see D5.6 Evaluation Report of service pilots), EOSC-hub Thematic Services (see D7.2 First report on Thematic Service architecture and software integration), EOSC Competence Centers (see D8.1 Report on progress, achievements and plans of the Competence Centres) and EOSC use cases identified through the EOSC Portal (see the EOSC-hub Community Requirements Database).

EOSC-hub already identified a considerable number of building blocks per service category and completed the technical specifications of the most relevant. However, we consider fundamental involvement of other relevant stakeholders in this work to have a real impact on the research world. For example, we think that including other technical experts in refining technical specifications and finding consensus around them to be essential. For this reason, we started a process to share our approach and collect feedback. The first step was a webinar where we presented this work[5], followed by a formal collection of feedback and we are planning to organise a workshop by the end of this year involving the largest expected EOSC user groups.

Finally, EOSC-hub intends to propose the contribution to the definition of the EOSC technical architecture described in this document, including the related approach to define the EOSC technical specification for building blocks, to the EOSC Architecture WG for its adoption in the wider EOSC environment, as soon as this WG will be fully operative. EOSC-hub would also like to collaborate with the WG on further refining the proposed architecture taking into account requirements and suggestions from the largest possible set of service providers and user communities.

[5] https://www.eosc-hub.eu/events/eosc-hub-proposal-eosc-technical-architecture

1 Introduction

The aim of the work presented in this document is increasing the added value provided by EOSC and fostering its uptake through the definition of a reference Technical Architecture for EOSC that facilitates access to services, lower barriers to integrate and composes services and promotes the usage of services between adjacent communities. This is achieved identifying key technical functions, named building blocks in the rest of the document, for each of the EOSC service category (Federation, Access Enabling, Common and Thematic) and defining related technical specifications that include an **high-level architecture, suggested EOSC standards and APIs and interoperability guidelines**. In this way, EOSC 'compliant' services will offer well-established and documented interfaces for usage and integration, based on well-known standard or APIs, facilitating:

- their exploitation from user communities willing to create new scientific services that could rely on well-established and documented interfaces for the integration. An example of exploitation of EOSC services is when a community creates a new scientific workflow re-using EOSC federation and common services, like AAI, accounting, Cloud orchestrator and/or data management solutions.
- the combined usage of EOSC services, indeed the adoption of well-known standards and interfaces will very likely reduce the cost to integrate services. For example, two accounting infrastructures can be made easily interoperable if they use the same standard usage record format, in such case accounting data extracted from them can be merged and presented in a unique view. Another example is about data processing and data management services implementing compliant interfaces that enable a jointly usage by a thematic service.

As a consequence, less mature or small scientific communities can leverage on EOSC services for a series of IT functions and focus on their scientific work, access to scientific services will be open to new communities thanks to the documented interfaces and new scientific workflows can be created combining existing applications.

This deliverable focuses on the EOSC Technical Architecture. The work on standard roadmaps mentioned in the title will be reported on the D10.1 and D10.2 EOSC-hub Technical Roadmap v1 and v2.

The document is organized as follows:

- Section 2 describes the landscape around the work on the EOSC Technical Architecture definition showing the connection of our work with the EC EOSC implementation roadmap and the EOSC Governance. Information on the past work on defining the EOSC Technical Architecture is also provided.
- Section 3 describes EOSC service categories and their organisation into EOSC portfolios as a basis for defining the architecture.
- Section 4 introduces the concept of end-to-end composition of services and how EOSC-hub is fostering the interoperability of services.
- Section 5 presents the proposed EOSC Technical Architecture describing a hierarchical structure and a functional view. A common approach to identify and detail each building block is depicted.

- Section 6 shows how requirements collected by several EOSC use cases are driving this work.
- Section 7 maps our outcomes with the objectives of the EOSC Architecture WG.
- Sections 8 and 9 present examples of EOSC technical specification for federation and common services.
- Finally, section 10 draws conclusions and describes next steps.

2 Landscape

This section describes the landscape around the work on the EOSC Technical Architecture definition showing the connection of this EOSC-hub effort with the EC EOSC implementation roadmap and the EOSC Governance.

Furthermore, a brief analysis of the past work on this topic is presented, notably the contribution for the definition of the EOSC Architecture of the EOSCpilot project.

2.1 EC Implementation Roadmap

In March 2018, the European Commission released a Commission Staff Working Document on the implementation roadmap for the European Open Science Cloud (EOSC)[6] where a model was proposed that *describes a pan-European federation of data infrastructures built around a federating core and providing access to a wide range of publicly funded services supplied at national, regional and institutional levels, and to complementary commercial services. The model includes six actions lines: (a) architecture, (b) data, (c) services, (d) access and interfaces, (e) rules and (f) governance*.

Figure 1. The six action lines of the EOSC implementation roadmap.

The Architecture action line, the most relevant for the work described in this document, foresees the creation of a *federation of existing and planned research data infrastructures, adding a soft overlay to connect them and making them operate as one seamless European research data infrastructure*. The EOSC architecture should *comprise a federating core and a variety of federated research data infrastructures committed to providing services as part of the EOSC* offered through the EOSC hub. Services are categorised in horizontal services, *such as a portal, authentication and authorisation and security services, allowing users to access the computing, data and services of pan-European and disciplinary research data infrastructures*, and in generic (also called common) or

[6] Commission Staff Working Document - Implementation roadmap for the European Open Science Cloud: https://ec.europa.eu/research/openscience/pdf/swd_2018_83_f1_staff_working_paper_en.pdf

thematic services, *for data storage, management and analytics, simulation and visualisation, distributed computing, etc*. Furthermore, the document underlines that EOSC services should come from existing European data infrastructures. The EOSC *hub would relay the resources and the services of data infrastructures funded at EU, national and regional level*, and should be accessible from a central portal (e.g. the EOSC Portal), *EOSC would provide a single, coherent access channel to EOSC services at European level that meets researchers' needs for data sharing, management and computing*. Finally, it is mentioned that the federation of the services should be gradual and *based on simple guidelines consistent with existing good practices*.

A first attempt on defining the EOSC Federating Core is presented in the EOSC-hub briefing paper "EOSC Federating Core Governance and Sustainability"[7], while the organisation of the EOSC services in portfolios according to the service categories depicted in the Commission Staff Working Document is described in the EOSC-hub D2.6 "First Service roadmap, service portfolio and service catalogue"[8]. Both concepts are shortly summarised in section 3. The work presented in this document leverages such definitions and builds the technical architecture for EOSC on top of them.

2.2 EOSC Architecture Working Group

The EOSC Governance[9], to fully implement its structure, defined five working groups (WGs)[10] to ensure a community-sourced approach to the current challenges of the EOSC:

- **Landscape:** Mapping of the existing research infrastructures which are candidates to be part of the EOSC federation;
- **FAIR:** Implementing the FAIR data principles by defining the corresponding requirements for the development of EOSC services, in order to foster cross-disciplinary interoperability;
- **Architecture:** Defining the technical framework required to enable and sustain an evolving EOSC federation of systems;
- **Rules of participation:** Designing the Rules of Participation that shall define the rights, obligations governing EOSC transactions between EOSC users, providers and operators;
- **Sustainability:** Providing a set of recommendations concerning the implementation of an operational, scalable and sustainable EOSC federation after 2020.

The activity of the Architecture WG is strictly related to the work EOSC-hub is doing on defining the EOSC technical architecture. This can be deduced by its main objective[11]: [the WG] *proposes the technical framework required to enable and sustain an evolving EOSC federation of systems. Such a technical framework may include standards, APIs and protocols that will facilitate interoperable services delivered by diverse providers*.

[7] https://documents.egi.eu/document/3479
[8] https://documents.egi.eu/document/3470
[9] https://www.eoscsecretariat.eu/eosc-governance
[10] https://www.eoscsecretariat.eu/eosc-working-groups
[11] https://www.eoscsecretariat.eu/working-groups/architecture-working-group

The need for defining an EOSC interoperability layer to enable the end-to-end composition of services delivered by various providers is mentioned in the WG mandate. To achieve this objective the WG is intended to describe and/or define:

- *EOSC core services and their interfaces;*
- *EOSC open source APIs for reuse by thematic services;*
- *EOSC portal components and federated catalogues of service offerings;*
- *the EOSC data description standards;*
- *Standards and best practices necessary to ensure the evolution of EOSC and the widening of its user base to the industry and the public sectors.*

The EOSC-hub work on technical architecture has been shaped with the same objectives in mind; this is clearly described in section 7 where we mapped the outcomes of our work to the Architecture WG objectives. Through its representative in the WG, EOSC-hub is expected to propose the EOSC technical architecture described in this document and the related approach to defining EOSC standard building blocks (see later for details) to the WG for its adoption in the wider EOSC environment. EOSC-hub would also like to collaborate with the WG on further refining the proposed architecture, taking into account requirements and suggestions from the largest possible set of service providers and user communities.

2.3 Past work on the EOSC Technical Architecture

2.3.1 EOSC Pilot Service Architecture

The EOSCpilot project was the first initiative that worked on the definition of the EOSC technical architecture. The architecture model described in the deliverable EOSC-Pilot D5.4 Final EOSC Service Architecture[12] is based on 47 classes of services *needed to develop and operate a system suitable to support the EOSC mission and goal*. These classes of services were organised in architecture from both a user and a functional perspective.

In the functional architecture, services were split into five categories:

- ***Front-end services**, for implementing the part of the overall service with which users will interact directly, namely portals or APIs;*
- ***Security & Trust**, aimed at guaranteeing that the overall system (and the services) operate securely and according to standards;*
- ***Open Science, Data Management, Analytics**, aimed at providing their users with user- and open-science-friendly facilities, enabling users to focus on science tasks;*
- ***EOSC System Governance & Management**, dedicated to supporting the operation and management of the overall EOSC System;*
- ***Compute & Cloud Platforms**, offering generalist resources like virtual machines and containers as well as network transport connectivity. In addition, all the platforms and software that do not belong to the other categories falls here.*

The following figure shows an overall view of the function architecture proposed by EOSCpilot.

[12] https://eoscpilot.eu/content/d54-final-eosc-service-architecture

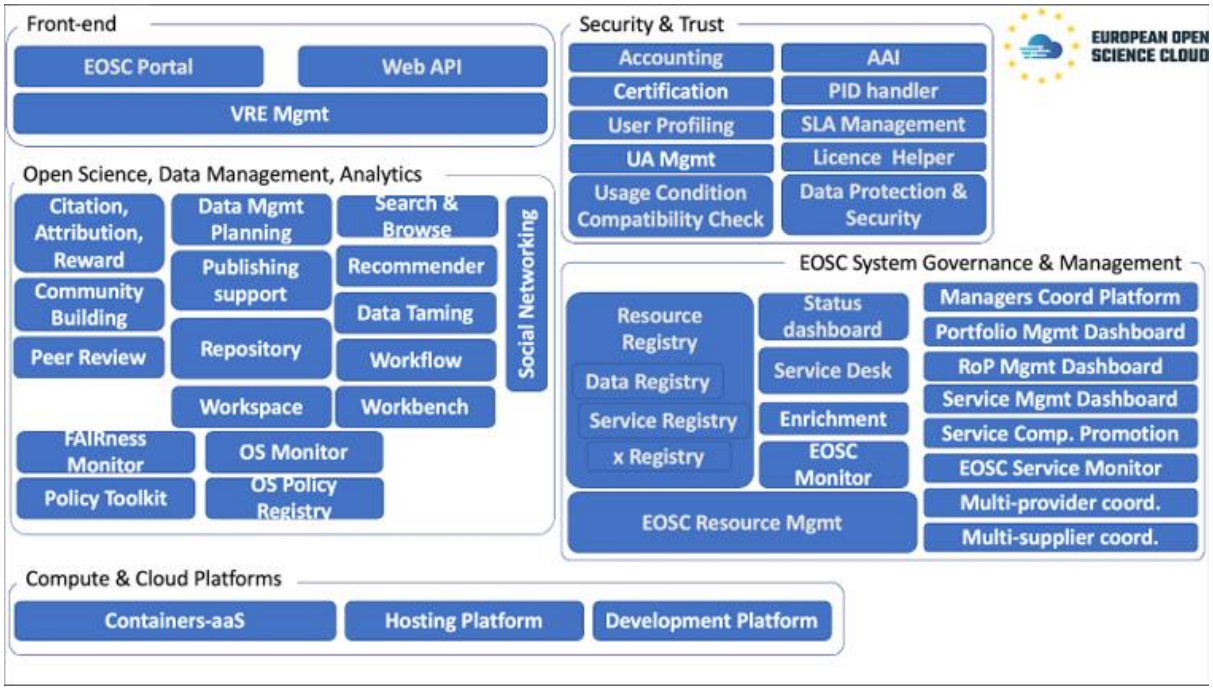


Figure 2. EOSCpilot - proposed EOSC function architecture

There are many similarities between the EOSCpilot functional architecture, and the work described in this document. Indeed, the EOSC technical architecture described in the next sections is also based on a classification of the services according to their functions, like the one proposed by EOSCpilot. Leveraging on the EOSCpilot experience, EOSC-hub refined the service classification, also taking into account the Commission Staff Working Document on the implementation roadmap for the EOSC, and went further ahead:

1. clarifying the interactions between the different service classes (the first level on the hierarchy in the proposed EOSC technical architecture).
2. defining an approach to create EOSC technical specifications and interoperability guidelines for each service/feature offered by EOSC.
3. proposing EOSC technical specifications and interoperability guidelines for key EOSC services/features.

2.3.2 EOSC-hub Technical Architecture v1

The first EOSC-hub deliverable on EOSC technical architecture[13] established the groundwork for the work described in this document. It identified the EOSC service types and their relationships and introduced the concept of end-to-end compositions of services describing the effort of the project on fostering the service interoperability both promoting the adoption of well-known standard and with ad-hoc integration activities driven by user requirements.

The document also presented the procedures to extend the EOSC service offer, federating/on boarding new services, and a deep analysis on the main standards, APIs and protocols used by the

[13] EOSC-hub D10.3 Technical Architecture v1: https://documents.egi.eu/public/ShowDocument?docid=3417

services belonging to a specific technical area. Such analysis was a needed preparatory phase to start the definition of the EOSC technical specifications.

The work presented in this document further analysed the concept of end-to-end compositions of services and built architecture that, from one side, leverages on the defined EOSC service types and relationships, and, from the other side, fosters service interoperability providing specifications and guidelines to develop and integrate services.

3 The EOSC Portfolios and the EOSC Federating Core

The proposed EOSC Technical Architecture is based on the different classes of EOSC services and on their interactions. Then, an introduction on such service categories and on their functions and relationships is necessary before describing the architecture.

As depicted in Figure 3, EOSC services are organised in two service portfolios:

- **EOSC Service Portfolio:** the external services which EOSC-hub either provides from its partners or onboards from the community to contribute to the larger portfolio of researcher-benefitting services within EOSC. The EOSC Service Portfolio contains:
    - **Thematic services:** community-specific capabilities including research core data, data products, scientific software, and pipelines. Examples of thematic services are data resources and software tools to access, study and compare the data; data brokering services tailored to the needs of specific scientific communities;
    - **Common services:** they provide generic capabilities usable by any science discipline each supporting aspects of the data lifecycle from creation to processing, analysis, preservation, access and reuse. Examples of services belonging to this category are multi-disciplinary services for data discovery, processing, workflow management and orchestration, data management, etc.
- **Hub Portfolio:** the internal services contributing to the federating core of EOSC, both for internal operation of the EOSC Hub and to offer as components to be integrated into the services of the EOSC Service Portfolio. They enable the other EOSC elements to deliver (greater) value to researchers across Europe. They can be further split in
    - **Access-enabling services:** delivering features allowing customers to easily exploit EOSC resources such as discovery, ordering and workflow enabling services
    - **Federation services:** needed to operate the EOSC e.g. a common helpdesk, accounting information gathering, monitoring

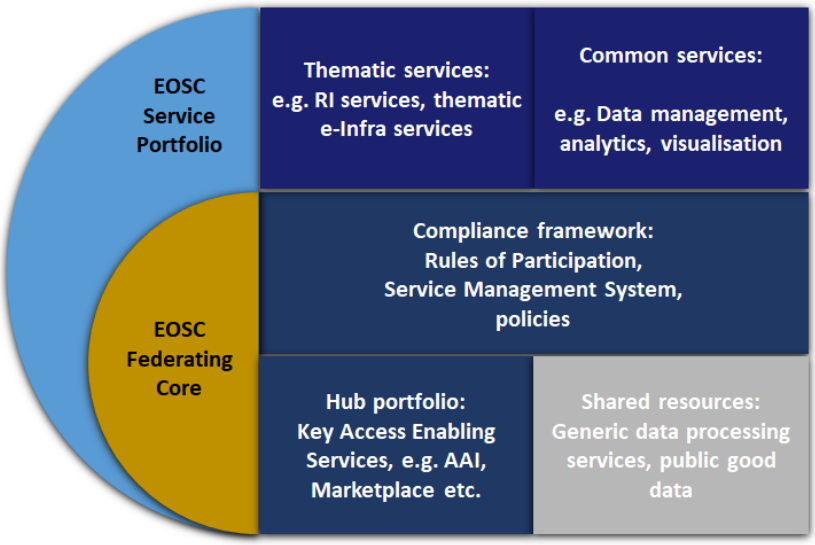


Figure 3. EOSC Service Portfolios and EOSC Federating Core

Thematic services can be integrated with the services in the Hub portfolio to facilitate the users' access (e.g. the EOSC Portal and Marketplace) or to avoid re-implementing basic features, like authentication and authorisation, accounting, monitoring, etc. They can also adopt common services that already address some of their technical needs. Common services can also leverage on services of the Hub portfolio to deliver some of their functions. The integration of thematic, common, access enabling, and federation services can be fostered through a large adoption of open and standard interfaces.

The Hub Portfolio is one of the key elements of the **EOSC Federating Core** together with the Compliance framework - made of Rules of Participation, EOSC Service Management System and other policies - and the Shared resources, a set of generic data processing and managing services, commodity services, compute and storage resources and public good data managed and offered centrally by EOSC. More information about the EOSC Federating Core and the EOSC service portfolios are available in the EOSC-hub briefing paper "EOSC Federating Core Governance and Sustainability"[14] and in the EOSC-hub D2.6 "First Service roadmap, service portfolio and service catalogue"[15].

In the context of the Technical Architecture, we then discuss three categories relevant to our technical work:

- Federation and access enabling services
- Common services
- Thematic Services

Federation and access enabling are key services needed to operate the EOSC (e.g. the EOSC Portal or the accounting infrastructure). Common services offer add-value features on top of EOSC resources (computing, storage, etc.) and can be reused by a multitude of other services. Thematic services implement discipline specific features and are provided directly by scientific communities.

[14] https://documents.egi.eu/document/3479
[15] https://documents.egi.eu/document/3470

4 Service Composability

The end-to-end composition of the services can be considered one of the most important added values provided by EOSC. Indeed, the service composability would allow EOSC service providers and users to select various services offered by EOSC and compose them according to their needs to create added-value solutions for research.

Typical service combinations are:

- A thematic service adopts some EOSC federation services to implement basic features (AAI, monitoring, accounting).
- A thematic service adopts common services that provide features to better exploit compute, storage and data resources including those offered by distributed infrastructures.
- An EOSC user creates new scientific workflows integrating, for example, a data repository and some analytics services together.

The adoption of standard interfaces makes some of the services of the EOSC service catalogues already interoperable, these sub-classes of composable services need to be identified and made accessible through the EOSC Portal. Furthermore, in response to newly emerging needs from communities, other services can be made interoperable through integration activities. EOSC should provide technical guidelines (in terms of suggests EOSC standards and APIs) and technical support to both integrate services and facilitate the combined usage of (already) interoperable services.

Enabling the service composability would allow to lower the barriers for developers of the thematic services to reuse common, federation or access enabling services to implement basic features (AAI, accounting, monitoring, etc.) and exploiting in the best way compute, storage and data resources. Indeed, they, from one side, can focus on working on increasing the scientific added value of their services, and from the other side, rely on well-established and EOSC-compliant services for implementing the basic features. A large part of these reusable services will come from the experiences of the main European e-infrastructures and other relevant initiatives (such as those involved in the project, EGI, EUDAT and INDIGO-DataCloud).

4.1 Fostering the service interoperability

Interoperability is a key concept to enable the end-to-end composition of services in EOSC. EOSC-hub is already working on fostering the interoperability in EOSC in a dual way:

- identifying services that can already work together because they support the same standards and/or interfaces. An example is shown in Figure 4 where the CREODIAS DATA HUB[16], one of the DIAS platforms funded by the EC for handling Copernicus Data, is working with the Sentinel Hub[17], a tool that uses Copernicus data to create maps. The services are already interoperable because they both support the OGC WMS standard[18]. Therefore,

[16] https://marketplace.eosc-portal.eu/services/cloudferro-data-collections-catalog
[17] https://marketplace.eosc-portal.eu/services/sentinel-hub
[18] https://www.opengeospatial.org/standards/wms

specific integration work is not needed and EOSC should provide technical support to the community willing to exploit these services in a combined manner.

- integrating federating, common and thematic services according to identified users' requirements. An example is shown in Figure 5, the DODAS analytics services has been integrated with the EGI Federated Cloud, to use its cloud resources, and the EGI Check-in services, to implement the AAI. In such a case, the integration required some development. The solution made of these integrated services is used by the CMS community. The solution can be reused by other communities without any further integration work.

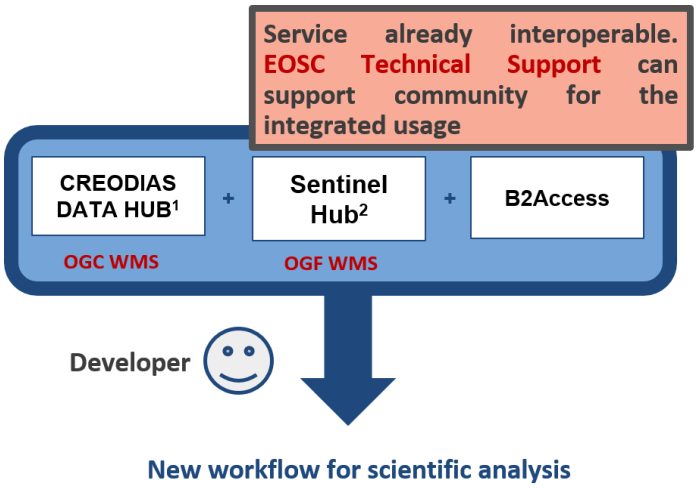


Figure 4. Examples of composition of services supporting the same standard (e.g. OGC WMS).

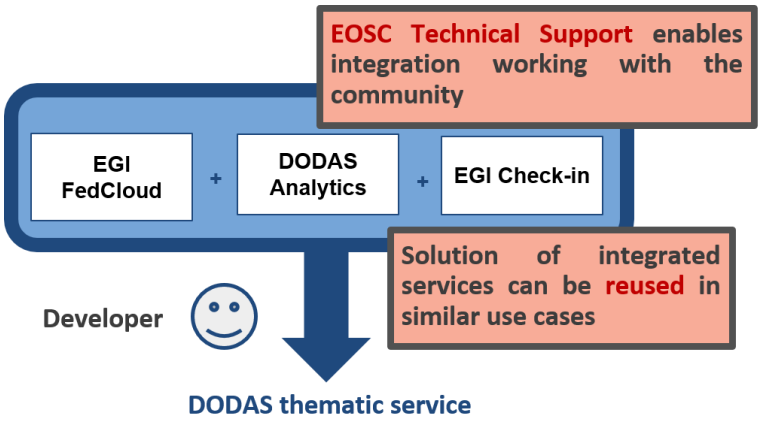


Figure 5. Examples of composition of services obtained through an integration activity.

Both approaches allow identifying and extending the set of services that can work together /can be composed. These solutions can be offered to all the EOSC users that should be able to recognise and reuse them, also thanks to the technical support offered by EOSC.

This work to identify and make services interoperable would be easier if EOSC interfaces for integration, possibly based on well-known standards and API, would have been already available

together with clear instructions or procedures to allow services to interact and work together. Interfaces and instructions to make interoperable a given service or a technical feature can be called **EOSC interoperability guidelines**. The full set of such interoperability guidelines can be considered a valid implementation of the EOSC interoperability layer mentioned in the mandate of the EOSC Architecture WG. The model of technical architecture presented in section 5 is based on these concepts and is thought to foster the service interoperability and, then, the end-to-end composition of services.

The EOSC interoperability guidelines that are being defined in the context of this work will take into account and will be based on existing community practices, well-known standards and interfaces. All relevant EOSC stakeholders (communities, e-infrastructures, etc) should be able to describe and promote their standards and practices for their inclusion in the EOSC guidelines. Adoption of these guidelines by providers will not be forced, making them mandatory, but should be a natural consequence of the advantages, for a service, generated by being compliant such as offering access through well-known interfaces, low cost to interoperate with other EOSC services, etc.

4.2 Federating thematic services in the EOSC

When a provider of a thematic service decides to join the EOSC, it should be able to consult the EOSC service offer and, consequently, decide (or not) to adopt/integrate EOSC services. EOSC will present to the provider its service portfolios (the Hub portfolio and the EOSC portfolio) as a sort of à-la-carte menu from which the provider can, first of all, understand the benefits of adopting a certain service and, then, can assess the technical feasibility and the related cost of the integration. This technical assessment will be possible only if the selected EOSC services offer well-established and documented interfaces for the integration, the EOSC interoperability guidelines. Furthermore, the integration cost will be very-likely reduced if such interfaces are based on well-known standard and interfaces.

After the analysis of the EOSC service offer, the provider of the thematic service can decide which EOSC services to adopt. This choice can be very different for each provider; some providers can decide to join the EOSC with no integration except listing their services in the service catalogue and/or marketplace. Others can opt for a tighter integration, adopting services from both the Hub and EOSC portfolios. For example thematic services can decide to adopt only the Marketplace and the AAI from the Hub Portfolio, while another thematic service can use a plethora of services from both portfolios (Marketplace, AAI, Accounting, and Monitoring and Helpdesk from the Hub Portfolio, a cloud orchestrator and a data management tool from the EOSC Service Portfolio).

5 Defining the EOSC Technical Architecture

This section details the process used by EOSC-hub to define the EOSC Technical Architecture. The architecture presented is a reference architecture where service categories, building blocks and related interfaces are identified. It focuses on the concepts of service interoperability and composition introduced in the previous section, fostering the definition and the adoption of EOSC standards and interfaces. EOSC-hub is proposing an implementation of this reference architecture as described in section 5.2.

5.1 Reference Architecture

As stated above, the EOSC Technical Architecture presented in this section is reference architecture. In the field of software architecture or enterprise architecture, reference architecture provides a template solution for architecture for a particular domain. It includes a common vocabulary with which to discuss implementations, often with the aim of stressing commonalities. A reference architecture often consists of a list of functions, some indication of their interfaces (or APIs) and interactions with each other and with functions located outside of the scope of the reference architecture[19].

Reference architectures can be defined at different levels of abstraction, in the context of EOSC, EOSC-hub decided to work at the infrastructure/technical level. As part of this work, we are also defining a common vocabulary that can be used to define both existing services and those joining EOSC catalogue in the future. The architecture includes functions, interfaces, APIs and standards as technical concepts, with the final aim of fostering interoperability and, ultimately, service composability. It is based on a hierarchical structure with three levels. These are:

1. Category (the service categories introduced earlier).
2. Functional categories within the main category.
3. Individual building blocks usable in fulfilling these functions.

An overview is seen in Figure 6.

[19] S. Angelov, P. Grefen and D. Greefhorst, "A classification of software reference architectures: Analyzing their success and effectiveness," 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Cambridge, 2009, pp. 141-150.
doi: 10.1109/WICSA.2009.5290800
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5290800&isnumber=5290660

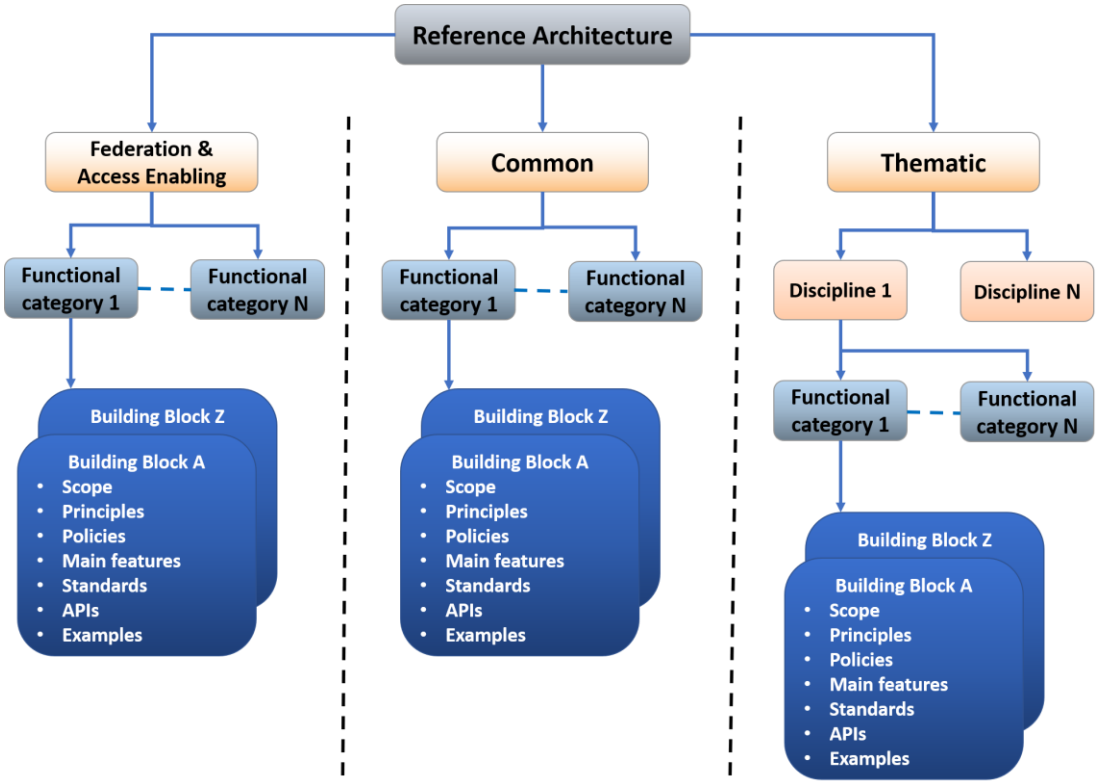


Figure 6. Hierarchical structure in the EOSC reference architecture.

The subdivision in categories allows differentiating services according to their function within EOSC: The top level categories (Federation & Access Enabling, Common and Thematic) have already been introduced. The second level of the hierarchy introduces the functional categories that groups technical functions to facilitate their identification. To take an example, within Federation and Access enabling services, we may see Authentication & Authorisation or Monitoring as functional categories. In the case of thematic services, the functional categories are identified per scientific discipline.

Beneath this, we see the individual building blocks that implement technical functions. To continue the example, within the Authentication & Authorisation functional category, we see the AAI building block.

The reference architecture described in this section gives flexibility on defining the second and third level of the hierarchy, functional categories and building blocks. Section 5.2 presents the implementation of this reference architecture proposed by EOSC-hub where functional categories and building blocks are started to be defined for each service category.

5.1.1 Approach to define building blocks

EOSC-hub is working on defining the building blocks of the architecture for each service type and has specified a common approach to complete this task. It foresees the identification of the main **building blocks/technical functions** in each service category and, for each of those, defining a **technical specification** that includes an **high-level architecture, suggested EOSC standards and APIs**

and interoperability guidelines. This method would allow providers offering services implementing the technical function of a given building block to be compliant with the related EOSC technical specification. As a consequence, thanks to the provided guidelines, interoperability between services offering the same technical function(s) and following the EOSC specifications will be easier to achieve. Hence sets of services implementing the same building block and compliant with the EOSC specification can be made able to work together with less effort, to deliver a given technical function in the EOSC environment. Examples of these service families can be AAI services compliant with the AARC blueprint architecture and guidelines or monitoring and/or accounting systems able to exchange/share information and provide integrated views to the EOSC customers and service providers. This approach is tailored to the varied environment seen in EOSC, where many solutions to satisfy a given technical requirement already exist.

Furthermore, the definition of EOSC standards and APIs along with related interoperability guidelines for each of the identified building blocks will foster the end-to-end composition of services. Being compliant with a specification for a given building block, would allow a service to interoperate with other services offering the same function (as described above) and, conversely, building blocks/services offering different technical functions can interoperate thanks to the EOSC interfaces, described in the technical specification (e.g. it would be easier for a thematic service integrating a common service if clear interoperability guidelines are available).

EOSC interoperability specifications are not intended to be mandatory but being compliant with them would be an added value for services. Indeed, they could interoperate with other services with less effort and reduced cost. Therefore, providers willing to expand their user base by making their services composable will be inclined to support such specifications.

In this approach, identifying building blocks and the respective technical specifications could be a complex and long work, so the consortium has agreed to follow an iterative approach, starting from the functions that are more requested by the EOSC use cases[20]. Technical specifications, initially prepared by the technical experts within the EOSC-hub project, should also be iteratively improved, collecting feedback by external people with expertise in the area and involving them in the maintenance and evolution of such specifications. The same is true for the list of building blocks: they will evolve and change in the future, adding/removing functions depending on the user requirements and on the projects/service providers that may join the EOSC in the future. This will be an ongoing, continuous activity that should be continued within EOSC after the end of the project.

[20] EOSC-hub is taking into account in this work requirements collected from EOSC Pilot Scientific Demonstrator (see D5.6 Evaluation Report of service pilots), EOSC-hub Thematic Services (see D7.2 First report on Thematic Service architecture and software integration), EOSC Competence Centers (see D8.1 Report on progress, achievements and plans of the Competence Centres) and EOSC use cases identified through the EOSC Portal (see the EOSC-hub Community Requirements Database).

5.1.2 Technical Specification template

We have defined a template to collect information about each of the identified building blocks and define a technical specification, regardless of the service category they belong to. It is structured as follows:

- Introduction: short description of the building block highlighting its main functions.
- High-level Service Architecture: reference architecture of the building block, highlighting the interfaces towards the other building blocks. It does not refer to any specific service.
- Adopted Standard: list with references of the main adopted standards and protocols/API.
- Interoperability guidelines: describe how services implementing this building block can be made interoperable.
- Examples of solutions implementing this specification: list of already available services that are compliant with this specification.

The complete template is available at the following address https://documents.egi.eu/public/ShowDocument?docid=3529.

5.2 Proposed EOSC Technical Architecture

Figure 7 shows the functional view of the proposed EOSC technical architecture, as implementation of the reference architecture described in the previous section, where the interactions between services belonging to different categories are highlighted.

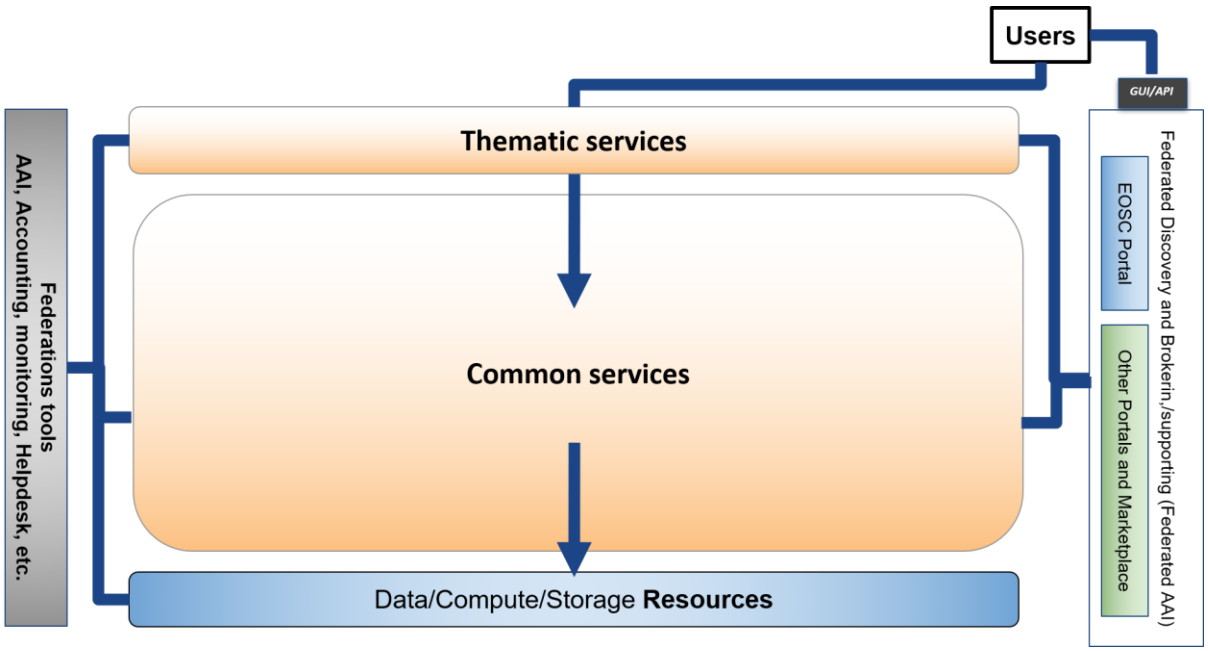


Figure 7.EOSC Technical Architecture - Functional view.

EOSC users can exploit EOSC Thematic and Common services directly or through the GUI or API of an access enabling services like the EOSC Portal (or other portals and Marketplaces). Thematic services can leverage Common services for added value features on top of data, compute and storage resources. Federation tools support all these services providing basic features like

authentication and authorisation, accounting, monitoring, etc. Pledged shared resources centrally managed by EOSC, including both commodity services and service capacity, are part of the Resources and complement other EOSC resources directly managed by other service providers.

This functional view will be better detailed in the following sections of the document with information on the already identified and defined building blocks per service category.

5.2.1 EOSC Access Enabling and Federation services

In the Access Enabling and Federation service categories, **a building block is any key access-enabling and federation function needed to operate the EOSC**. Services offering these features according to the EOSC specification could be onboarded on the Hub service portfolio described in section 3.

We already identified an initial list of building blocks for this category, leveraging the experiences from some of the largest European e-infrastructures that are involved in the project. In this initial phase, we have a one-to-one mapping between functional categories and building blocks, this may change in the future.

This list is detailed in the table below.

Functional categories/Building blocks	Short description
EOSC Portal	The EOSC Portal provides a European-level delivery channel, connecting the demand-side (the EOSC Customers) and the supply-side (the EOSC Providers) to allow researchers to conduct their work in a collaborative, open and cost-efficient way for the benefit of society and the public at large. In particular it delivers the following functions: ● Enable different kinds of users, with different skills and interests, to discover, access, use and reuse a broad spectrum of EOSC Resources (services, datasets, software, support, training, consultancy, etc.) for advanced data-driven research ● Support interdisciplinary research and facilitate Resource discovery and access at the institutional and inter-institutional level ● Allow researchers and institutions to focus on value creation through sharing and reuse as opposed to duplicating Resources and increase excellence of research and European competitiveness ● Improve the provisioning of access to integrated and composable products and services from the EOSC Catalogue ● Facilitate the composition of services and products to support multi-disciplinary science for example with high-level community-specific interfaces for running workflows involving EOSC services ● Help Providers gain additional insight into potential User groups outside their traditional constituencies ● Give Providers the possibility to offer Resources under homogeneous terms of use, acceptable use policies, and in

	different configuration options, so that Users are guided in the choice. **Use case.** The Portal is particularly relevant to support on-demand access to EOSC through Business-to-User (B2U) and Business-to-Business (B2B) transactions. ● B2U is applicable for consumer-oriented Resources appealing to a large potential User pool. B2U transactions will address the digital needs of individual researchers and short- and medium-term research projects. Because of the potential large user base, B2U transactions will be most suitable for those Resources supporting automated or semi-automated provisioning, a short acquisition process, requiring a low-level of specialisation, and which can be easily compared and chosen without requiring expert support. ● On the other hand, B2B applies predominantly to the acquisition of bespoke solutions and/or of large quantities of EOSC Resources involving potentially multiple Providers. B2B suits the needs of research performing organisations and research infrastructures which need to cater for the long-term needs of a large pool of end users. The EOSC Portal Concept 2.0[21] provides extensive information on potential use cases and a participatory model for resource providers, which are provided with the choice of selecting different EOSC participation levels.
AAI	The EOSC AAI aims to enable seamless access to multiple research data and services in EOSC in a secure and user-friendly way. It also provides authorisation management for access control. It is based on the AARC blueprint architecture. The EOSC AAI follows the architectural and policy recommendations defined in the AARC project[22]. As such, it enables interoperability across different Service Provider(SP)-Identity Provider(IdP)-Proxy services, each of which acts as a bridge between the community-managed proxies (termed Community AAIs) managing the researchers' identity and the generic services offered by Research Infrastructures and e-Infrastructures (termed R/e-Infrastructures or Infrastructures). This is the "community-first" approach to the AARC Blueprint Architecture[23], which enables researchers to sign in with their community identity via their Community AAI. Community-specific services are connected to a single Community AAI, while Infrastructure Services are connected to a single Infrastructure Proxy. Lastly, generic services may be connected to more than one Community AAI. Each Community AAI in turn serves as a bridge between external identity providers and the proxies to the e-infrastructure services. Specifically, Community AAIs connect to eduGAIN as service providers but act as identity providers from the

[21] https://wiki.eosc-hub.eu/display/EOSC/EOSC+Portal
[22] https://aarc-community.org
[23] https://aarc-project.eu/guidelines/aarc-g045/

	services point of view, thereby allowing users to use their credentials from their home organisations. Complementary to this, users without an account on a federated institutional Identity Provider are still able to use social media or other external authentication providers for accessing services. Research communities can leverage the EOSC AAI services for managing their users and their respective roles and other authorisation-related information. At the same time, the adoption of standards and open technologies, including SAML 2.0, OpenID Connect, OAuth 2.0 and X.509v3, facilitates interoperability and integration with the existing AAIs of other e-Infrastructures and research communities. **Use Cases.** Access to all EOSC shared resources and access enabling services (e.g. the Portal, the Helpdesk, EOSC data and compute and storage resource tier) will require federated authentication and authorisation.
Helpdesk	The helpdesk is the tool that supports Incident and Service Request Management to restore normal/agreed service operation within the agreed time after the occurrence of an incident, and to respond to user service requests. The service works as a unified ticketing system, by connecting individual providers' helpdesks to the central helpdesk instance, offering a standalone service interface. **Use case.** The helpdesk tool is necessary to support Incident and Service Request Management of the resources provided by EOSC. The helpdesk can be implemented as a distributed platform linking together the helpdesks of suppliers offering resources to EOSC. The linking of existing helpdesks allows streamlining of support processes involving multiple suppliers, and in particular facilitates the work of the support teams that, through linking, are able to use existing in-house tools.
Monitoring	Monitoring provides the capability to check the status of service end-point interfaces and aggregate such information for the production of service reports. In particular, it should provide a scalable framework for monitoring the status, availability and reliability of endpoints. It provides monitoring of services, visualisation of their status, dashboard interfacing, notification and generation of availability and reliability reports. Third parties can gather monitoring data from the system through a complete API. **Use case.** Monitoring information supports Service Report Management, and is consumed to produce Service Reports, i.e. the documents that provide the details of the performance of a service against the service targets defined in service level agreements (SLAs) – often based on key performance indicators (KPIs). Typical users are the EOSC service suppliers.
Accounting	Accounting is about collecting, aggregating, storing and displaying EOSC resource usage data produced by the providers participating in EOSC, for example from the providers of Shared Resources. It gathers usage information from the individual resource providers and aggregates it

	centrally in a secure, GDPR-compliant manner. Accounting is necessary for providing control over resource consumption by the funders and reduces the overhead of each separate resource provider defining accounting information models, architecture and setup. Accounting is a key service of the EOSC federating core that will support its business models and provides transparency on which resources are being used. The correlation of usage data to service identifiers, scientific product identifiers and user identifiers, supports the development of metrics that relate scientific impact to the extent a researcher and/or project has been embracing open science practices. **Use case.** Accounting of resource usage is required for any EOSC customers (e.g. platform operators and research infrastructure managers) to enable aggregated information on usage of scientific products and services used from the EOSC portfolio.
Federated Configuration Management DataBase (CMDB)	A configuration database is a database used by an organisation to store information about hardware and software assets (commonly referred to as Configuration Items). This database acts as a data warehouse for the organisation and also stores information regarding the relationship between its assets. The CMDB provides a means of understanding the organisation's critical assets and their relationships. At a federation or EOSC-level, it is a database drawing selected configuration information from provider CMDBs, which is needed at the EOSC or federation level. **Use case.** The availability of an EOSC CMDB is relevant to EOSC shared resource suppliers and is requested by the IT configuration management process. It allows the management of the provision of services owned and managed by the EOSC governance. It is envisaged that the management of resources published in EOSC just for the purpose of improving their discoverability, will be delegated to the respective providers and will not be registered in an EOSC CMDB.
Order management	Order management is a process allowing the portal operators to handle orders received through the EOSC Portal. It implements interfaces towards service provider order management processes to support orders that should not be centrally processed in EOSC. **Use case.** Managing orders from the EOSC Portal.
Operations Portal	The Operations Portal refers to the set of control dashboards that support the work of EOSC infrastructure managers in charge of supervising the overall status, allocation and accessibility of the EOSC shared resources. It provides central operations management of federated resources. The Operations Portal offers a portfolio of management tools to support communications, customer relationship management, infrastructure oversight, and metrics gathering. **Use case.** The Operations Portal can support multiple service management activities like incident management and order management if used as a back-office tool of the EOSC Portal.
Service Portfolio	The Service Portfolio Management Tool (SPMT) allows lifecycle

Management Tool	management of the services provided through EOSC. SPMT allows providers to capture, store and maintain key information about their services, and to easily publish that data into an EOSC-mandated service catalogue, such that that hosted on EOSC-Portal. **Use case.** The tool is used by providers with one or more services which they which to deliver through EOSC. It simplifies their management of information about the services, simplifies delivery of this information to those managing onboarding to EOSC, and this simplifies the publishing of these services in a public catalogue.
Collaboration software & platforms	Tools needed to operate a 'Hub' or federating core for EOSC. These include collaborative documentation and document creation and management systems, issue management for task tracking and communication tools to manage remote collaborations. **Use case.** Collaborations between EOSC users and/or service providers.
Security monitoring	Provide features to monitor the security of the EOSC services and resources. **Use case.** Identify security threats in the EOSC.
Messaging	A real-time messaging service allowing to exchange messages between independent applications. **Use case.** Enabling asynchronous communication between EOSC services.
Software quality assurance	A tool allowing to deliver quality software for the EOSC consumption. The software is compiled, validated and distributed following the Software Provisioning Process (SWPP), where the Quality Criteria (QC) definition sets the minimum quality requirements for acceptance. The growing number of software components currently existing to support EOSC infrastructure favours the adoption of automated solutions instead of manual-based validation mechanisms. **Use case.** Automated validation of software quality.

Technical specifications for all these building blocks are under preparation and will be published for feedback[24] as soon as they are ready. The maturity level of the technical specifications of these building blocks varies; an example of an already mature specification is that for AAI, which is described later in the document. We intend to have mature specifications for all the building blocks of this category by the end of 2019.

The EOSC Portal is as special case within this category. It is currently being further enhanced and developed by a large collaboration that includes EOSC-hub, OpenAIRE Advance and key partners from the former eInfraCentral project. More information is available in the EOSC Portal concept

[24] Feedback will be collected through a public consultation that will be launched in the EOSC-hub website.

paper.[25] The outcomes of this collaboration will be adopted by this work to technically specify the EOSC Portal. Some details about the status of this activity are reported in section 5.4.

5.2.2 EOSC Common services

In the Common services, **a building block is a technical function that offers added value on top of EOSC resources (computing, storage, etc.) and that can be adopted by multiple thematic services**. Examples of building blocks for this category are Infrastructure as a Service (IaaS) Virtual Machine (VM)/Container management, Cloud Orchestration, metadata management, making scientific artefacts FAIR, etc. A building block in the Common Services category can be implemented and, then, offered by one or more common services.

In this category the number of relevant building blocks can be huge, so we must split the work into sub-areas or functional categories. We used the different technical areas EOSC-hub is working on as a basic functional division to start the process of identifying the building blocks:

- HTC/HPC Compute
- Cloud Compute (including Containerisation and Orchestration)
- PaaS Solutions
- Data Platforms for Processing
- Data Publishing and Open Data
- Data Preservation/Curation/Provenance
- Metadata Management and Data Discovery
- Workflow management, user interfaces and Data analytics

Other functional categories could be added by other initiatives according to their expertise. For example, OpenAIRE suggested the addition of a 'Scholarly Communication' category and proposed building blocks for this area during the last EOSC-hub technical workshop in Amsterdam[26].

How the work was split into technical functions is shown in the following figure. Within the Common Services category, a set of functional categories/technical areas where depicted (only four areas are shown as a sample) to simplify the identification of the building-blocks.

[25] https://wiki.eosc-hub.eu/display/EOSC/EOSC+Portal
[26] https://indico.egi.eu/indico/event/4675/overview

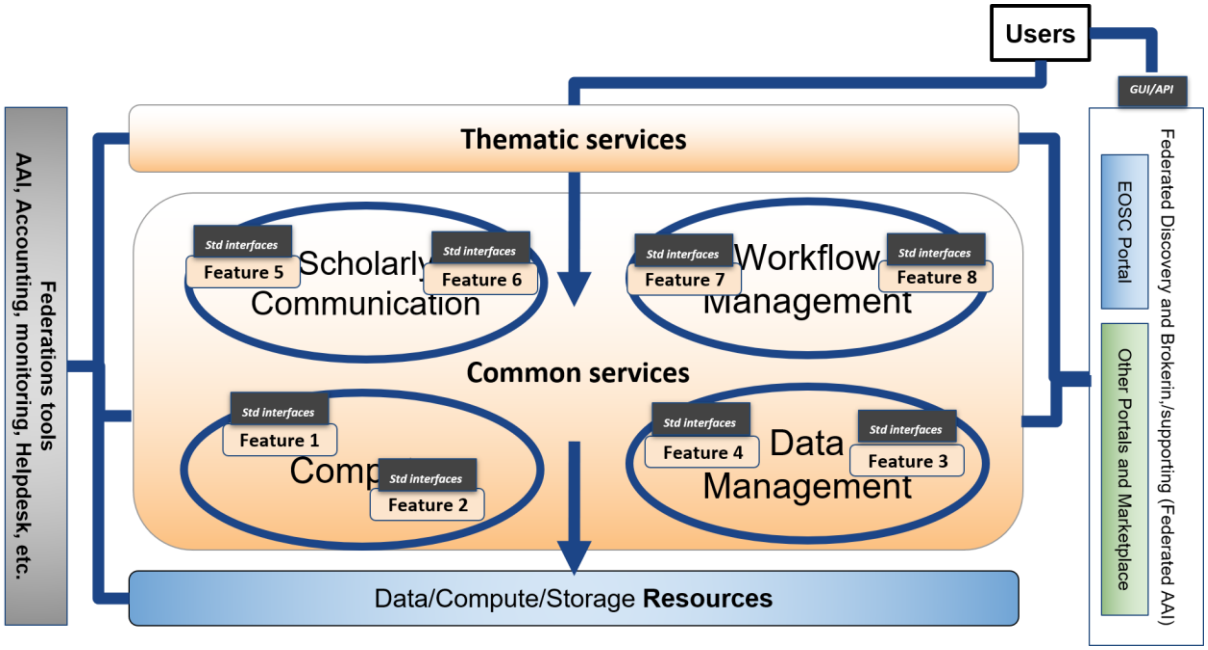


Figure 8. EOSC Technical Architecture. Building blocks per functional category.

For the Common Services, we agreed to prioritise the preparation of the technical specifications for the building blocks that are more relevant to users according to the use case analysis. The current list of identified building blocks, organised per technical area, is in the following table.

Functional categories	Building blocks
HTC/HPC Compute	• Multitenant job submission • Multitenant container-based job submission • HTC / HPC clusters on demand
Cloud Compute (including Containerisation and Orchestration)	• IaaS: VM Management • IaaS: Orchestration • IaaS: Containers
PaaS Solutions	• PaaS Solution for Cloud service automation and federation of hybrid Cloud resources
Data Platforms for Processing	• Transparent data processing using POSIX in distributed and hybrid cloud environments including Docker, Kubernetes and Jupyter (Notebooks and Hub) • Data Ingestion and transfer for processing in hybrid cloud environment • Metadata Management in processing workflows • QoS based data access optimization and tight integration with preservation services • Authorization based on attributes from IdP • Results sharing and experiment repeatability

	● Distribution of software for the processing tasks
Data Publishing and Open Data	● Data Repository
Data Preservation/Curation/Provenance	● Data Preservation ● Tracking of provenance metadata ● Data Curation
Metadata Management and Data Discovery	● Data Discovery and Access ● Metadata cataloguing and indexing ● Annotation service ● Cloud based IoT Platforms interoperability
Workflow management and user interfaces and Data analytics	● Portals ● Big data analytics ● ML/DL analytics services
Scholarly Communication	● Data Management Plans ● Digital Preservation ● Overlay platforms: Peer-review ● Anonymization ● Aggregator ● Broker ● Entity Registry ● Metadata validation ● Annotation ● Usage stats ● VRE: RI Services for experiments

Examples of completed technical specification for building blocks belonging to the Common Services are presented later in this document.

The picture 9 shows how the functional view of the EOSC technical architecture will appear when the first set of building blocks and the related technical specifications are well defined. Thematic services could easily exploit building blocks offered by common services through the EOSC standard interfaces (purple arrows in the figure). Also, common services implementing such building blocks can be made interoperable in an easier way thanks to the EOSC standard interface (red arrows in the figure) offering a combined usage to the thematic services. In this scenario, service composability would be easier to attain, and the cost of integration work will be reduced with respect to the current situation.

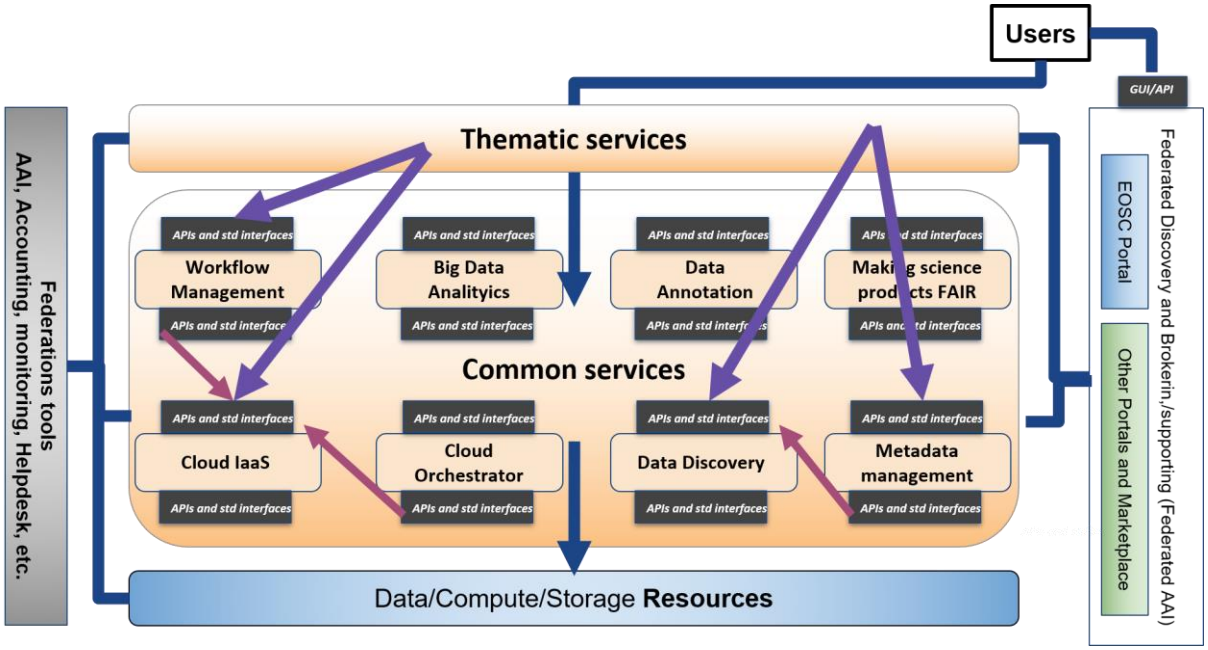


Figure 9. EOSC Technical Architecture. Interactions between thematic and common services.

5.2.3 EOSC Thematic services

As previously written, we want to apply the same process for Thematic services, to identify and create technical specifications for their building blocks. In this category, **a building block is a technical function that is discipline-oriented and that can be reused in multiple services within one thematic domain.**

Discipline oriented building blocks need to be identified and specified by experts of the related disciplines. Then, EOSC-hub will start the work of detailing this category with the communities participating in the project. However, community-oriented projects need to be involved to further enhance this activity.

The following picture shows how the EOSC technical architecture will appear when the first set of building blocks (and the related technical specifications) for the Thematic Services will have been identified.

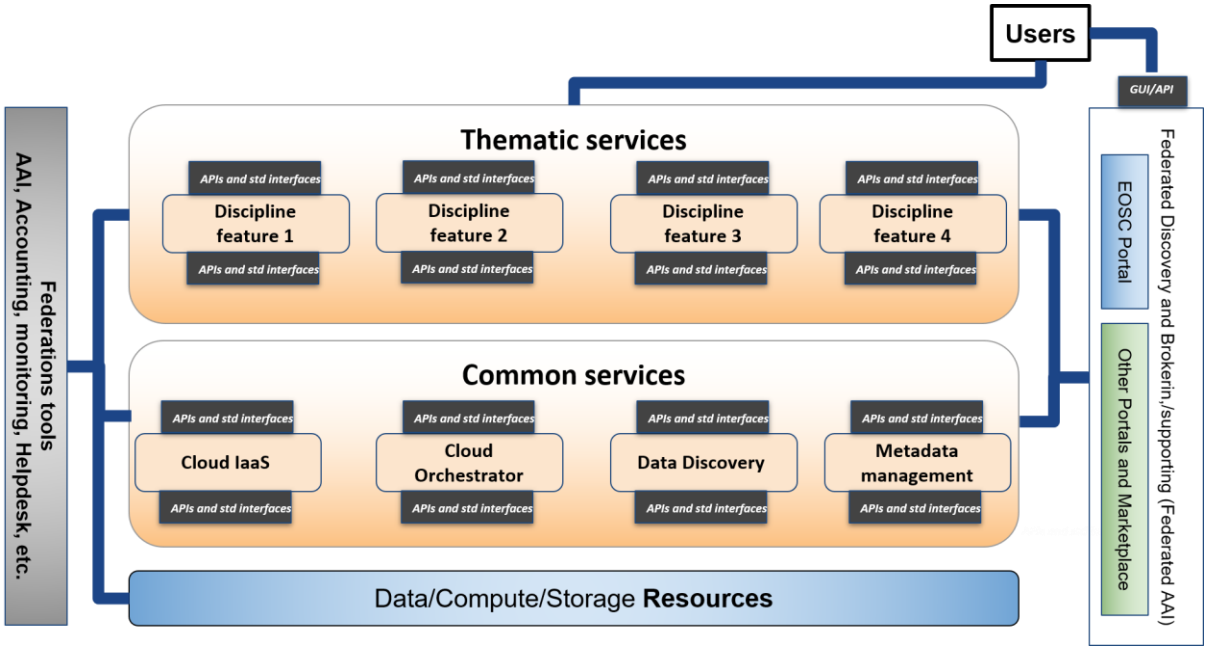


Figure 10. EOSC Technical Architecture. Building blocks for thematic services.

5.2.4 EOSC Portal

The EOSC Portal is intended to become the *single, coherent access channel to EOSC services at European level that meets researchers' needs for data sharing, management and computing* mentioned in European Commission Staff document introduced in section 2. It is currently being further enhanced and developed by a large collaboration that includes EOSC-hub, OpenAIRE Advance and key partners of the eInfraCentral project in the context of the EOSC Portal collaboration agreement (until December 2019). More information is available in the EOSC Portal concept paper.[27] The outcomes of this collaboration will be adopted by this work to technically specify the EOSC Portal. From 2020, EOSC-hub and OpenAIRE Advance are expected to continue the collaboration involving the new project that will be funded under the INFRAEOSC-06 "Enhancing the EOSC portal and connecting thematic clouds" call[28].

The following section is an extract of the architecture section of the EOSC Portal concept paper.

5.2.4.1 Architecture

The EOSC Portal concept paper describes the EOSC Portal architecture with a number of internal components and dependencies from external services.

[27] https://wiki.eosc-hub.eu/display/EOSC/EOSC+Portal

[28] https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/infraeosc-06-2019-2020

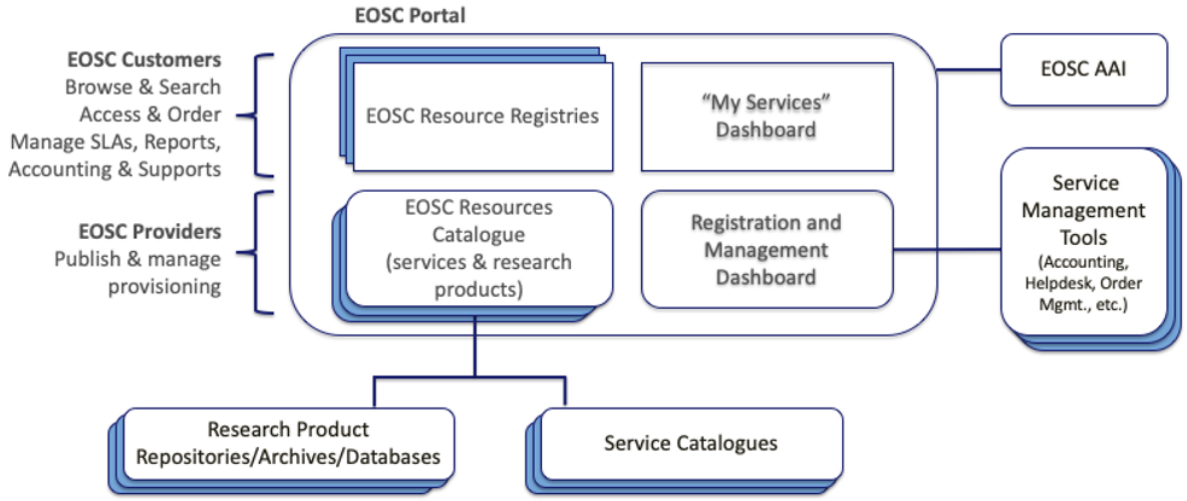


Figure 11. Internal components of the EOSC Portal and dependencies on external services

- **Internal components**
  - **EOSC Resource Registry:** *from an EOSC Customer point of view, browsing, searching, access and ordering will be possible through the EOSC Resource Registry. According to the EOSCpilot Glossary 1.0[29], the EOSC Resource Registry provides the descriptions of live / ready-to-use EOSC Resources offered by the EOSC System. Resources will include services and scientific products that are produced by scientists, like data, software, publications, tools and experiments. These are published for discovery and reuse with metadata and links to other products via dedicated sources, e.g. repositories, archives, databases. To facilitate their discovery, cross-discipline or thematic metadata aggregators are today available and widely used by scientists. The EOSC Portal will integrate with scientific product catalogues capable of serving the needs of researchers from different disciplines. Dedicated Registries will be possible in order to present the EOSC offer to specific Users groups.*
  - **EOSC Resource Catalogue:** *from a EOSC Provider point of view, the Catalogue comprises "the list of all live EOSC Resources that can be requested by EOSC System Users". Resources are described by metadata that is either directly inputted after successful validation by the Provider, or through APIs [...].*
  - The **My Services Dashboard** will allow EOSC users to manage the services they are ordered and accessed.
  - The **Registration & Management Dashboard** will provide capabilities supporting the Provider in the on-boarding and validation procedure, and additional functions for integrated service management of the contributed Resources within EOSC. The Additional capabilities for the Providers willing to in that choose a high-level partnership.
- **External dependencies.** The EOSC Portal will interact with external services, namely:
  - *The **EOSC AAI service**, conforming to the AARC blueprint architecture and operational guidelines, supporting: (1) uniform representation of unique Users*

[29] https://eoscpilot.eu/eosc-glossary

identifiers, (2) a standardised way of expressing group membership/role/information and Resource capabilities, (3) non-web browser based access, (4) delegation, (5) release of mandatory Users attributes according to the REFEDS research and scholarship entity category, (6) operational security, incident response and traceability - REFEDS Sirtfi, (7) privacy requirements for processing personal information following the GEANT Data Protection Code of Conduct, (8) rules and conditions that govern access to and use of Resources following the WISE Baseline Acceptable Use Policy, and (9) assurance information following the REFEDS assurance framework and IGTF/AARC assurance profiles.

- ***External catalogues, repositories, databases and archives**, providing metadata on services and other products (e.g. datasets, software, applications). Interoperability will be enforced with the adoption of the EOSC Catalogue Framework* (see the concept paper for more information).
- ***External tools for service management** (accounting repositories, helpdesk, order management tools etc.), which will exchange ticket information, usage information and order information that are managed externally. Interoperability will be ensured through a Service Integration and Access Management interoperability framework that is being developed by the EOSC-hub project.*

5.2.5 Architecture governance

Different governance models can be envisaged per the EOSC service typologies embedded in the architecture. Indeed, the influence of the EOSC governance on defining technical specifications for the different service types can vary from fully authority for federation and access enabling services to simple endorsement of the specifications provided by the communities for thematic services, while a hybrid approach can be foreseen for common services.

Recommendations on the governance models to apply to the identified service categories are expected from the EOSC Architecture WG.

6 Use cases to drive the identification and the specification of the building blocks

The presented EOSC Technical Architecture foresees the definition of a technical specification for each of the building blocks identified per service category. It appears clear that the total number of building blocks can become quite big, and then a way to identify and focus on the most relevant functions is needed.

The natural way to prioritise the building blocks is taking into account the user requirements. For this reason, we are using as references the analysis done on a multitude of use cases in the context of the EOSCpilot and EOSC-hub projects.

All the sources of information we are taking into account are listed in the following table. This list will be extended at any time according to the suggestions we will receive from other relevant EOSC stakeholders.

Sources	Description	References
EOSCpilot Science Demonstrators	EOSCpilot selected fifteen science demonstrators, across different scientific domains with the purpose of providing insight on technical and policy needs, and cross-infrastructure integration requirements, and to get indications on how the EOSC Service portfolio should be structured. Some examples of communities supported in the context of this activity are: Photon and Neutron, EPOS, PanCancer, Fusion, WLCG, LOFAR, etc.	D5.6 Evaluation Report of service pilots[30]
EOSC-hub Thematic Services	Mature thematic services (TRL8 or TRL9) from large communities federating in the EOSC and integrating several generic, federation and access-enabling services. Involved communities are: CLARIN.	D7.2 First report on Thematic Service architecture and software integration[31]

[30] https://eoscpilot.eu/content/d56-evaluation-report-service-pilots

[31] https://documents.egi.eu/document/3412

	CMS, ENES, GEOSS, OpenCOASTs, WeNMR, DARIAH, LifeWatch and several Earth Observation services (including the CREODIAS DIAS platform).	
EOSC-hub Competence Centres	Competence Centres design, integrate and disseminate new, community specific services and service platforms. Each Competence Centre (CC), fosters the use of advanced digital capabilities and resources of EOSC by early adopter research communities in order to support data- and computing-intensive science. Competence Centres are driven by well-established and mature research infrastructure or international scientific collaborations: ELIXIR, Fusion, Marine, EISCAT-3D, EPOS-ORFEUS, LOFAR, ICOS and Disaster Mitigation.	D8.1 Report on progress, achievements and plans of the Competence Centres[32]
Use cases identified through the EOSC Portal	Since the launch of the EOSC Portal in November 2018, many communities ordered EOSC services through the Service Catalogue and Marketplace. These communities were supported by the EOSC-hub technical support activity (T10.3) and the analysis of their use cases where stored in the	Community Requirements Database[33]

[32] https://documents.egi.eu/document/3485

[33] https://wiki.eosc-hub.eu/display/EOSC/Community+requirements+DB

	Community Requirement Database.	

Collected requirements are used to identify the most relevant building blocks but also to properly shape the related technical specifications. Indeed, user requirements give suggestions on the main functions a building block should deliver and on the most common integration scenarios. The latter would allow understanding which (standard/EOSC) interfaces a building block should implement to satisfy the needs of the largest number of use cases. The following figure shows hypothetical user requirements for the interoperability guidelines of various building blocks.

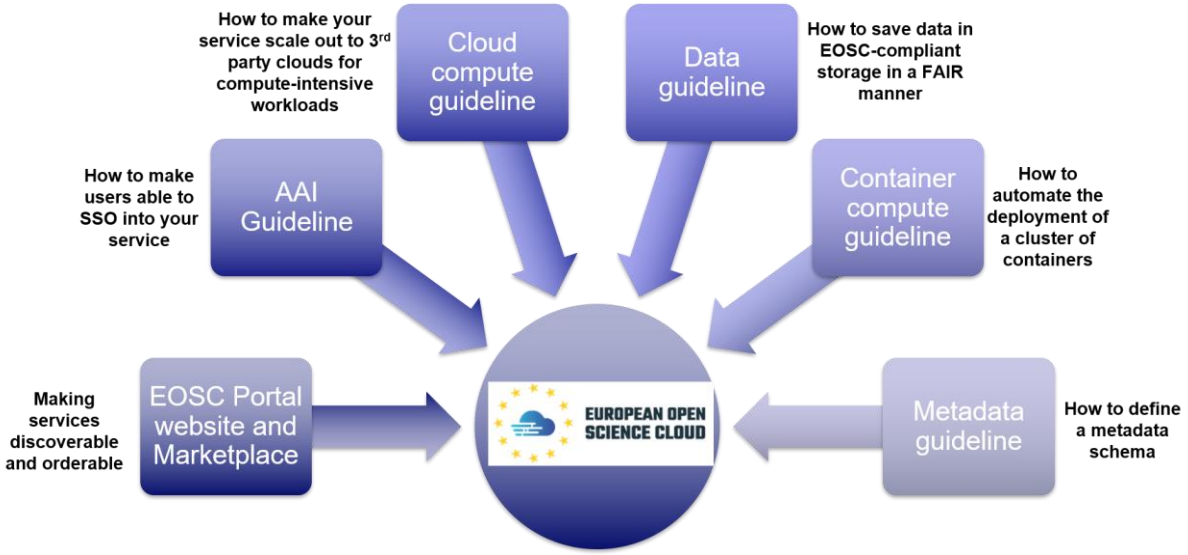


Figure 12. User requirements and interoperability guidelines for building blocks.

7 Relationship with the EOSC Architecture Working Group

As described in Section 2, the EOSC-hub work on the EOSC Technical Architecture is intended to become an important input for the activity of the EOSC Architecture Working Group and it has been shaped taking into account the WG mandate. As a result, the outputs of our work can become valuable for the WG and support it in achieving its objectives.

The following table shows the mapping between the Architecture WG sub-objectives and what EOSC-hub is expected to deliver.

EOSC WG objectives - The WG will describe and/or define:	**Outcome of the EOSC-hub activity on the EOSC technical architecture**
EOSC core services and their interfaces	● Definition of the EOSC Access Enabling and Federation services and interfaces
EOSC open source APIs for reuse by thematic services	● Interoperability guidelines for Common services (EOSC APIs and standards) ● Interoperability guidelines for Thematic services (EOSC APIs and standards)
EOSC portal components and federated catalogues of service offerings	● Outcomes of the collaboration between EOSC-hub, OpenAIRE and key partners from eInfraCentral on the EOSC Portal design and development
The EOSC data description standards	● To be described in the technical specification of the metadata management building block
Standards and best practices necessary to ensure the evolution of EOSC and the widening of its user base to the industry and the public sectors	● Interoperability guidelines for Common services (EOSC APIs and standards) ● Interoperability guidelines for Thematic services (EOSC APIs and standards)

EOSC-hub would like to establish a fruitful collaboration with the WG to further refine the proposed architecture taking into account requirements and suggestions from the largest possible set of service providers and user communities.

8 Technical Specifications for Federation services

This section (and the next one) present a complete snapshot of all the currently available technical specifications for Federation and Common Services proposed by EOSC-hub.

Updated versions of the given technical specifications will be available at https://wiki.eosc-hub.eu/pages/viewpage.action?pageId=52598376.

8.1 AAI

8.1.1 AAI

The EOSC AAI enables seamless access to research data and services in EOSC in a secure and user-friendly way.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
Security Assertion Markup Language (SAML) 2.0	OASIS standard for exchanging authentication and authorisation data between parties.	https://www.oasis-open.org/standards#samlv2.0
OAuth 2.0	Standard for authorisation that enables delegated access to server resources on behalf of a resource owner	"The OAuth 2.0 Authorization Framework", RFC 6749, https://www.rfc-editor.org/info/rfc6749
OpenID Connect 1.0	Identity layer on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner	“OpenID Connect Core 1.0”, https://openid.net/specs/openid-connect-core-1_0.html
X.509	ITU-T standard for a public key infrastructure (PKI), also known as PKIX (PKI X509)	"Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, https://www.rfc-editor.org/info/rfc5280

		"Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile", RFC 3820, https://www.rfc-editor.org/info/rfc3820
Lightweight Directory Access Protocol (LDAP)	Provides access to distributed directory services that act in accordance with X.500 data and service models.	https://tools.ietf.org/html/rfc4511

Protocol/API	**Short description**	**References**
OAuth 2.0 Token Introspection	Protocol that allows authorised protected resources to query the authorisation server for determining the set of metadata for a given OAuth2 token, including its current validity.	https://tools.ietf.org/html/rfc7662
OAuth 2.0 Token Exchange	Protocol for requesting and obtaining security tokens from OAuth 2.0 authorization servers, including security tokens employing impersonation and delegation.	https://tools.ietf.org/id/draft-ietf-oauth-token-exchange-14.html
OAuth 2.0 Device Authorization Grant	Enables OAuth 2.0 clients on input-constrained devices to obtain user authorisation for accessing protected resources without using an on-device user-agent.	https://tools.ietf.org/html/draft-ietf-oauth-device-flow-15
System for Cross-domain Identity Management (SCIM) 2.0	Open API for managing identities	SCIM: Core Schema, RFC7643, https://tools.ietf.org/html/rfc7643 SCIM: Protocol, RFC7644, https://tools.ietf.org/html/rfc7644 SCIM: Definitions, Overview, Concepts, and Requirements, RFC7642, https://tools.ietf.org/html/rfc7642

High-level Service Architecture

The EOSC AAI follows the architectural and policy recommendations defined in the AARC project. As such, it enables interoperability across different SP-IdP-Proxy services, each of which acts as a bridge between the community-managed proxies (termed Community AAIs) managing the researchers' identity and the generic services offered by Research Infrastructure and e-Infrastructures (termed R/e-Infrastructures or Infrastructures). This is the “community-first” approach to the AARC Blueprint Architecture, which enables researchers to sign in with their community identity via their Community AAI. A high-level view of the EOSC AAI is provided in Figure.

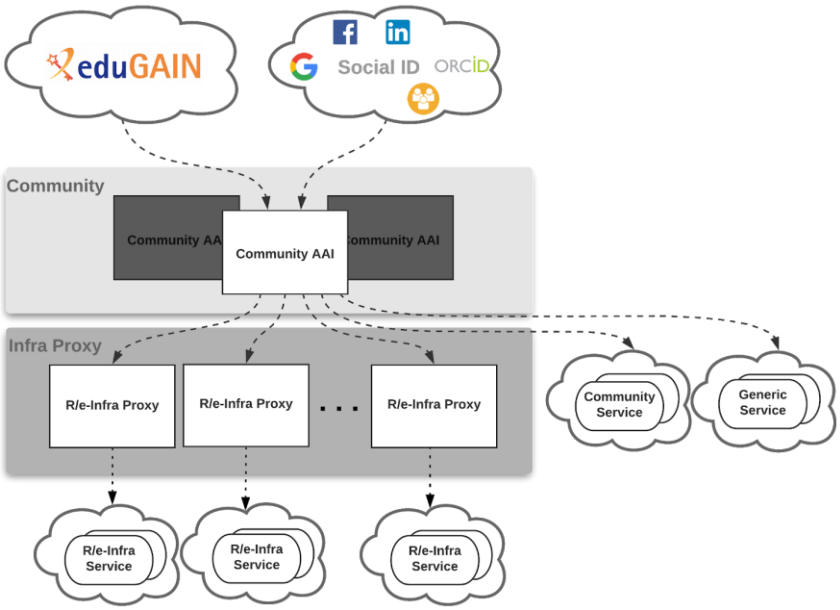


Figure 13: High level architecture diagram of the access to EOSC resources: A researcher’s perspective following the AARC Blueprint Architecture.

Community-specific services are connected to a single Community AAI, while Infrastructure Services are connected to a single Infrastructure Proxy. Lastly, generic services may be connected to more than one Community AAI. Each Community AAI in turn serves as a bridge between external identity providers and the proxies to the e-infrastructure services. Specifically, Community AAIs connect to eduGAIN as service providers but act as identity providers from the services point of view, thereby allowing users to use their credentials from their home organisations. Complementary to this, users without an account on a federated institutional Identity Provider are still able to use social media or other external authentication providers for accessing services.

Research communities can leverage the EOSC AAI services for managing their users and their respective roles and other authorisation-related information. At the same time, the adoption of standards and open technologies, including SAML 2.0, OpenID Connect, OAuth 2.0 and X.509v3, facilitates interoperability and integration with the existing AAIs of other e-Infrastructures and research communities. As shown in Figure 2, communities can allow different authentication

options for their members and, at the same time, enable access to all or a subset of the Infrastructures. It should be noted that this model also allows users to access resources as members of their home organisation. Being connected to multiple Community AAIs and the upstream institutional/social IdPs requires the Infra Proxies to properly support discovery for both community- and home organisation-based access scenarios.

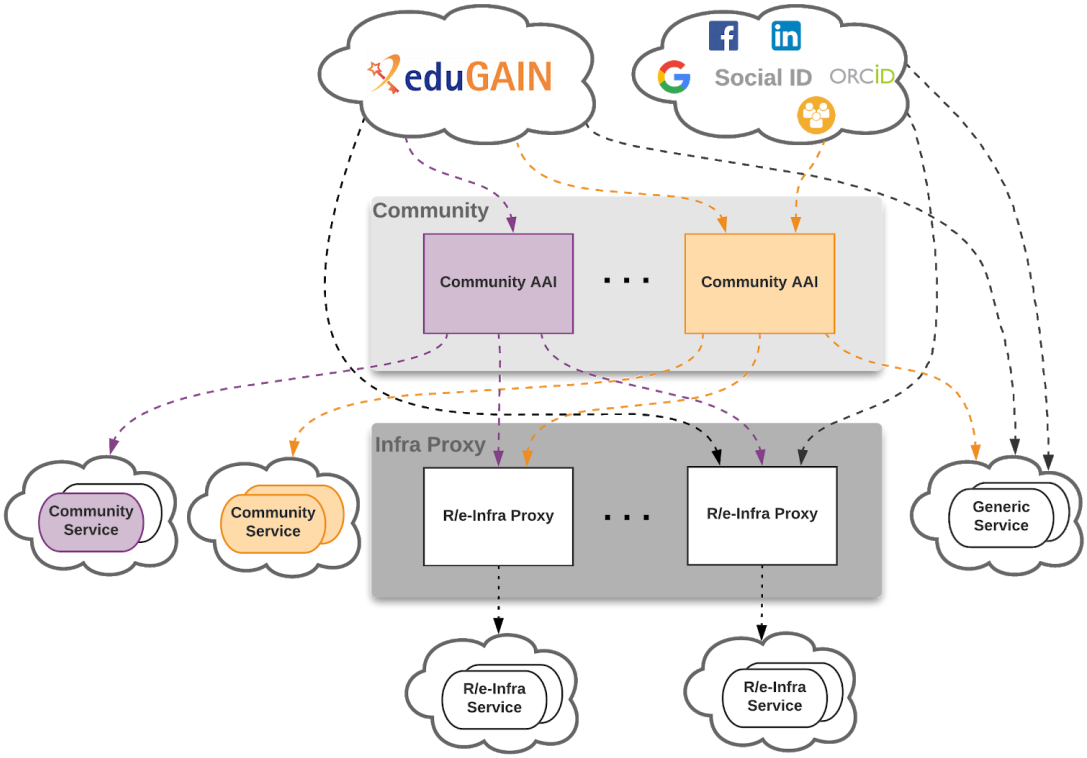


Figure 14: High level architecture diagram of the AAI architecture for accessing EOSC resources

Interoperability guidelines

Technical interoperability guidelines

- The attributes used to express user information should follow the REFEDS R&S attribute bundle, as defined in https://refeds.org/category/research-and-scholarship
- VO/group membership and role information, which is typically used by relying parties for authorisation purposes, should be expressed according to https://aarc-project.eu/guidelines/aarc-g002/
- Capabilities, which define the resources or child-resources a user is allowed to access, should be expressed according to https://aarc-project.eu/guidelines/aarc-g002/
- Affiliation information, including (i) the user's affiliation within their Home Organisation, such as a university, research institution or private company, and (ii) affiliation within the Community, such as cross-organisation collaborations, should be expressed according to https://aarc-project.eu/guidelines/aarc-g025/

- Assurance information used to express how much relying parties can trust the attribute assertions about the authenticating user should follow:
    - REFEDS Assurance framework (RAF)[34]
    - Guideline on the exchange of specific assurance information[35]
    - Guideline for evaluating the combined assurance of linked identities[36]
    - Guideline Expression of REFEDS RAF assurance components for identities derived from social media accounts[37]
    - Guidelines for expressing the freshness of affiliation information, as defined in https://aarc-project.eu/guidelines/aarc-g025/
- OAuth2 Authorisation servers should be able to validate tokens issued by other trusted Authorisation servers. Extending existing flows, such as the OAuth2 Token Exchange flow[38], will need to be considered for enabling the validation of such externally issued tokens.

Policy interoperability guidelines

For the EOSC AAI, compliance with the GÉANT Data Protection Code of Conduct version 1 (DPCoCo-v1)[39] is implicit, since it reflects the Data Protection Directive and means compliance with applicable European rules[40]. To explicitly declare compliance with DPCoCo-v1, the privacy notice of each EOSC AAI service should include a reference to DPCoCo-v1.

The entities of the EOSC AAI registered with eduGAIN should meet the Sirtfi requirements and express Sirtfi compliance in their metadata in order to facilitate coordinated response to security incidents across organisational boundaries.

To reduce the burden on the users and increase the likelihood that they will read the AUP as they access resources from multiple service and resource providers, the EOSC AAI services should adopt the WISE Baseline AUP model[41].

Examples of solutions implementing this specification

AAI services:

- B2ACCESS - https://www.eudat.eu/services/b2access
- Check-in - https://wiki.egi.eu/wiki/AAI_guide_for_SPs
- eduTEAMS - https://wiki.geant.org/display/eduTEAMS

[34] https://wiki.refeds.org/display/ASS/REFEDS+Assurance+Framework+ver+1.0

[35] https://aarc-project.eu/guidelines/aarc-g021/

[36] https://aarc-project.eu/guidelines/aarc-g031/

[37] https://aarc-project.eu/guidelines/aarc-g041/

[38] https://tools.ietf.org/html/draft-ietf-oauth-token-exchange-16

[39] https://wiki.refeds.org/download/attachments/1606087/GEANT_DP_CoCo_ver1.0.pdf?version=1&modificationDate=1450367740260&api=v2

[40] https://aarc-project.eu/guidelines/aarc-g040/

[41] https://wiki.geant.org/download/attachments/123766285/WISE-SCI-Baseline-AUP-V1.0.1-draft.pdf?version=1&modificationDate=1557297275149&api=v2

- INDIGO-IAM - https://indigo-iam.github.io/docs/v/current/

Identity and Access Management:

- Perun - https://perun-aai.org/documentation/technical-documentation
- Comanage - https://www.internet2.edu/products-services/trust-identity/comanage/
- HEXAA - https://hexaa.eu/

Token Translation Services:

- WaTTS - https://watts-prod.data.kit.edu/docs/user/rsp.html
- MasterPortal - https://wiki.nikhef.nl/grid/RCAuth.eu_MasterPortal_VOPortal_integration_guide
- RCauth.eu - https://wiki.nikhef.nl/grid/AARC_Pilot_-_RCAuth.eu#Self-Registration

Procedure to integrate a service with the EOSC Hub AAI

- B2ACCESS - https://www.eudat.eu/services/b2access
- Check-in - https://wiki.egi.eu/wiki/AAI_guide_for_SPs
- eduTEAMS - https://wiki.geant.org/display/eduTEAMS
- INDIGO-IAM - https://indigo-iam.github.io/docs/v/current/
- Perun - https://perun-aai.org/documentation/technical-documentation
- WaTTS - https://watts-prod.data.kit.edu/docs/user/rsp.html
- MasterPortal - https://wiki.nikhef.nl/grid/RCAuth.eu_MasterPortal_VOPortal_integration_guide
- RCauth.eu - https://wiki.nikhef.nl/grid/AARC_Pilot_-_RCAuth.eu#Self-Registration

8.2 Federation Tools

8.2.1 Helpdesk

The EOSC Helpdesk is the entry point and ticketing system/request tracker for issues concerning with the available EOSC services.

The features of the EOSC Helpdesk can be grouped by two target groups.

Main features offered to the user are:

- Creation of a ticket for any of the EOSC Services (Hub and EOSC Portfolios)
- Display all the tickets created by the owner
- Find a previously created ticket
- Notify the user of answers and changes to the tickets
- Access integrated with the EOSC Portal AAI system

Features offered to the Helpdesk Team should be:

- Notification when a new ticket is created

- Classification of the tickets
- Escalation of the tickets
- Creation of a new support unit[42] with assignation of an administrator role to specific users
- Management of incident or disruption of Hub services
- Interface for communicating with other service providers ticketing systems

First level support for EOSC integrated services as a service

Interface with a Known Errors Database and with a Change Management Database

Adopted standards

Coherence of information between systems and communication among them need a specified standard integration protocol.

Protocol/API	Short description	References
X-GUS protocol implemented over SOAP	SOAP method that allows communication between two helpdesk systems.	https://wiki.egi.eu/wiki/GGUS:SOAP_Interface_FAQ

Information about the structure and the semantics of the exchanged messages can be found in the referenced document.

High-level Service Architecture

The EOSC Helpdesk service, as part of the EOSC Federating Core, works as a unified ticketing system by managing the requests for different services or resources providers from a common standalone service.

The EOSC Helpdesk provides a 1st level support for all EOSC services and a dedicated 2nd level support for services in the EOSC Hub Portfolio (eg: Marketplace, AAI system, Monitoring, ...). Services of the EOSC Portfolio[43] can use EOSC Helpdesk choosing one of the following integration paths as shown in the figure above:

1 **Direct Usage**: Use directly the EOSC helpdesk as the ticketing system for the service (”EOSC Services support team” box in the dashed rectangle in the picture).
2 **Ticket Redirection**: Use the EOSC helpdesk only as a contact point to redirect the entry request for the specific service to a mailing list or 2nd level ticketing system (Common and Thematic

42 A support unit allows to identify tickets for a specific service. A dedicated team of supporters can be associated to a support unit.

43 Services in the EOSC Hub Portfolio support the operations of the EOSC and EOSC Portal. Services of the EOSC Portfolio are all the other EOSC services. For more details refer to the EOSC-hub D2.6: https://www.eosc-hub.eu/deliverable/d26-first-service-roadmap-service-portfolio-and-service-catalogue-approved-ec

services in the picture). In this case, the EOSC Helpdesk central service would simply redirect by e-mail the incoming tickets to the external system.

3 **Full Integration**: Integrate the service ticketing system with the EOSC helpdesk infrastructure to have full integration of the service Trouble Ticketing System (TTS) with the EOSC helpdesk ("Services in the EOSC Portfolio with integrated ticketing system" in the picture);

Services in the EOSC Portfolio should directly manage the 2nd level of support providing adequate human resources independently by the chosen integration path.

Each of the Helpdesk systems has its own database. The Helpdesk databases belonging to the EOSC Hub services or EOSC Portfolio services directly using the Helpdesk ("Direct usage" integration path) share the same infrastructure (marked with the dashed rectangle in the graph above). As a consequence, when the Helpdesk infrastructure is shared, the communication with the central Helpdesk database is direct.

For services with a Helpdesk in different infrastructures the integration towards EOSC is performed via interface and established protocols and APIs (in case of x-GUS and RT, the interface is SOAP), as indicated in the table below and shown as a yellow rectangle in the figure above. In the case a service maintains its own independent Helpdesk system, the user can directly access the external Helpdesk system. The Central system receiving tickets for services that chose the "Ticket redirection" integration path would redirect the incoming tickets by e-mail.

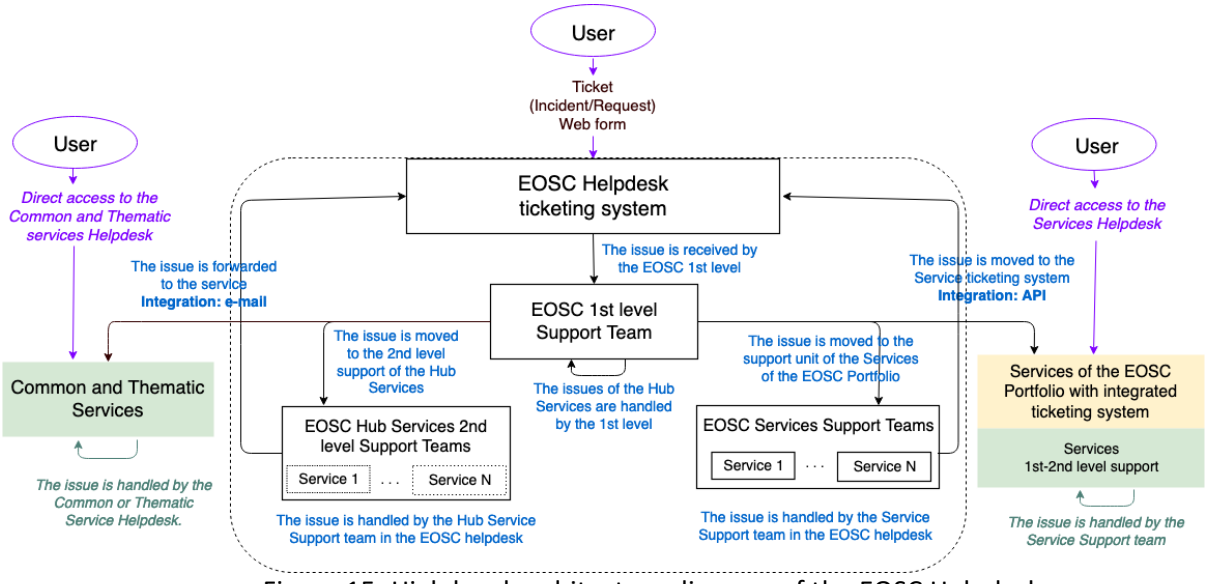


Figure 15: High level architecture diagram of the EOSC Helpdesk

Interoperability guidelines

For new services, there are three levels of interoperability corresponding to the three integration paths described above:

1 Direct Usage: Use directly the EOSC helpdesk as the ticketing system for the service. It implies the creation of accounts for the service owners and service responsible in order to receive the request and be able to answer them from the EOSC helpdesk system.

2 Ticket redirection: Use the EOSC helpdesk only as a contact point to **redirect** the entry request for the specific service to a mailing list or 2nd level ticketing system, without further integration.

3 Full integration: Integrate the service ticketing system with the EOSC helpdesk infrastructure to have **full integration** of the service TTS with the EOSC helpdesk (see next session). For this level of integration, the new services that should be made interoperable with the EOSC helpdesk will need to synchronize both endpoints via the interface specified and available at the main service infrastructure (next session will describe an integration solution through SOAP).

EOSC-hub developed a central helpdesk for EOSC using the XGUS technology. This helpdesk is currently offered through the EOSC Portal to providers during the service onboarding process. It can be found at: https://helpdesk.eosc-hub.eu and it is accessible through the EOSC Portal AAI.

XGUS has been used by EOSC-hub to implement the first-level Helpdesk that has been already integrated to EUDAT (RT) and EGI (GGUS) helpdesks (Full Integration option). New services joining the EOSC Portal can already use XGUS as helpdesk selecting one of the options described above.

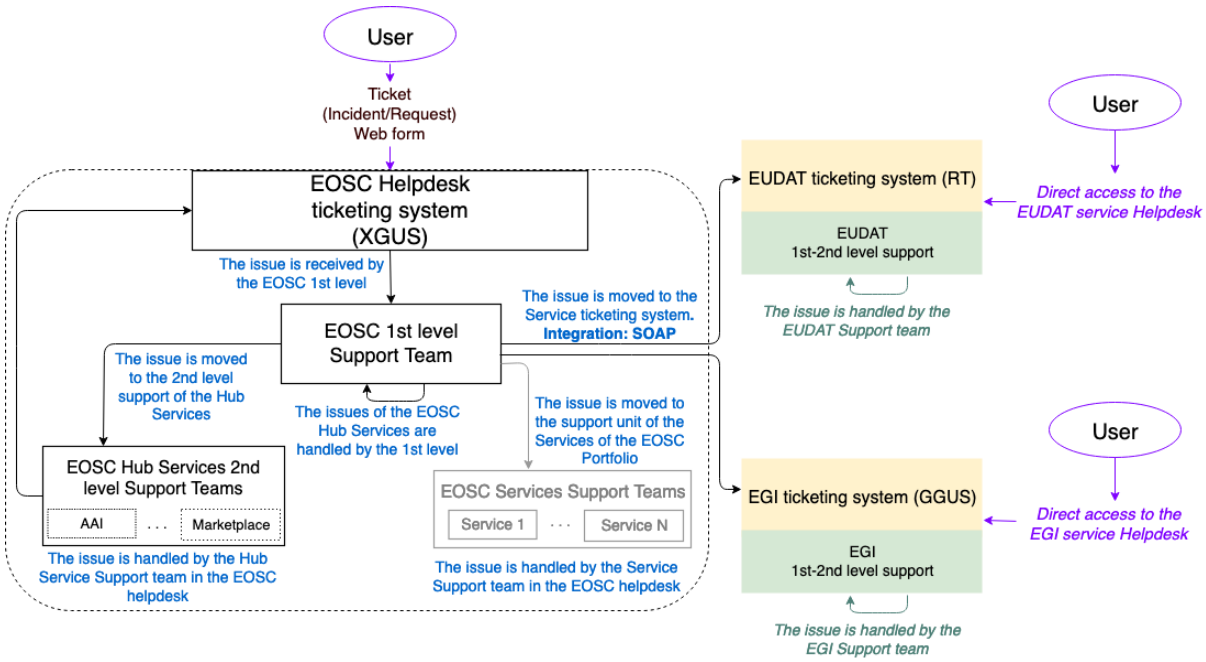


Figure 16: Current deployment of the EOSC Helpdesk

Currently, a service provider can decide to adopt the EOSC Helpdesk during the onboarding process. The request should be expressed filling in the Service Description Template.

The EOSC Portal onboarding team will register the request and put the service provider in contact with the Helpdesk team. The first two integration options (Direct Usage and Ticket Redirection of the EOSC helpdesk) will only require configurations on the central helpdesk. The XGUS support team will take care to gather the needed information from the service provider and to configure the central helpdesk accordingly.

If the provider select the third integration option (Full Integration), The SOAP interface provided by xGUS following the specification available at https://wiki.egi.eu/wiki/GGUS:SOAP_Interface_FAQ has to be developed.

Integration procedures are detailed in the last section of the document.

Examples of solutions implementing this specification

XGUS has been used by EOSC-hub to implement this specification. Details are available at https://xgus.scc.kit.edu/.

The EUDAT (RT) Helpdesk (http://helpdesk.eudat.eu/) system is already integrated with the EOSC-hub Helpdesk using the SOAP interface provided by xGUS. In this case, four RT scripts have been implemented in order to achieve a full integration:

- Owner change in RT is updated in xGUS
- Priority change in xGUS is updated in RT
- Status and priority changes in xGUS are updated in RT
- Status change in RT is updated in xGUS

The interface is described in the following WSDL, which can be located at: https://train-ars.ggus.eu/arsys/WSDL/public/train-ars/XGUS_EOSCHub This webservice uses a local authentication (username and password) in order to accept the communication.

Procedure to integrate a service with the EOSC Hub Helpdesk

The procedure to integrate a service in the EOSC helpdesk is the following, some steps are required only for the “Use directly the EOSC helpdesk” and “full integration” integration types.

1 Create in the xGUS Helpdesk service the Support Unit for the new service or infrastructure
2 Assign to the Support Unit the contact points to be notified when a request is assigned to the Support Unit
3 If the service owner/responsible wants to use xGUS as their own ticketing system (Use directly the EOSC helpdesk integration), it implies the creation of user accounts for the required people and the grant of the permissions to see and answer any request/incident assigned to their support unit.
4 Only for the services/infrastructures interested in the full integration, it implies to develop the required scripts in their own ticketing system to communicate with the xGUS soap interface, for this point the xGUS responsible will need to create a specific user/password for the service/infrastructure in order to make the connection.

8.2.2 Accounting

The EOSC Accounting service collects, stores, aggregates, and displays usage information of HTC compute, storage space, cloud VM and data set resources. This usage data is collected from the Resource Centres of the EOSC infrastructure.

Accounting information is gathered from distributed sensors into a central Accounting Repository where it is processed to generate summaries that are made available through an Accounting Portal. Depending on the use case the data may go via intermediate repositories that collate accounting data for particular regions, infrastructures or communities.

The Accounting Repository has a database backend and needs to ensure the exchange of accounting information with peer e-Infrastructures. The Accounting Portal receives and stores the resource centre, user, and user groups (e.g. Virtual Organisation/VO) level aggregated summaries generated by the Accounting Repository and provides views via a web portal. For example, by grouping resource centres in a country on specific time intervals a customized view can be generated and displayed. The databases are organized into resources record database (e.g. CPU, storage, dataset, etc), a User record database, and a topology database.

The main features of the EOSC Accounting can be grouped by two target groups.

Main features offered to the user are:

- Aggregated views of their usage wherever that usage occurred.
- Views that allow usage to be checked against allocation.

Features for resource providers:

- Provider-centric views of resource usage by users.
- Views that allow comparisons to be made between resource providers within and between regions and communities.

Adopted standards

Standard	Short description	References
APEL Grid Job Usage Record	Standard used within WLCG and EGI for exchanging grid accounting metrics for individual grid jobs.	https://wiki.egi.eu/wiki/APEL/MessageFormat#Job_Records
APEL Grid Summary Job Record	Standard used within WLCG and EGI for exchanging grid accounting metrics for aggregations of grid jobs.	https://wiki.egi.eu/wiki/APEL/MessageFormat#Summary_Job_Records
Cloud VM Usage Record	Standard adopted by the EGI Federated Cloud for exchanging cloud accounting metrics.	https://wiki.egi.eu/wiki/Federated_Cloud_Architecture#Cloud_Usage_Record

OGF StAR	Open Grid Forum standard for Storage Accounting Records, used to exchange storage space usage data.	http://cds.cern.ch/record/1452920/files/GFD.201.pdf
GOCDB Grid Topology	The GOCDB domain model closely resembles a subset of the GLUE 2 Grid model with additional entities.	https://wiki.egi.eu/w/images/d/d3/GOCDB5_Grid_Topology_Information_System.pdf
ARGO Messaging Service (AMS)	A Publish/Subscribe Service, which implements the Google PubSub protocol. It provides an HTTP API that enables Users/Systems to implement message-oriented service using the Publish/Subscribe Model over plain HTTP.	http://argoeu.github.io/messaging/v1/

High-level Service Architecture

Describe the architecture (commented diagram) of the building block highlighting the interfaces towards the other services.

The architecture should be generic. Please, do not refer to specific service.

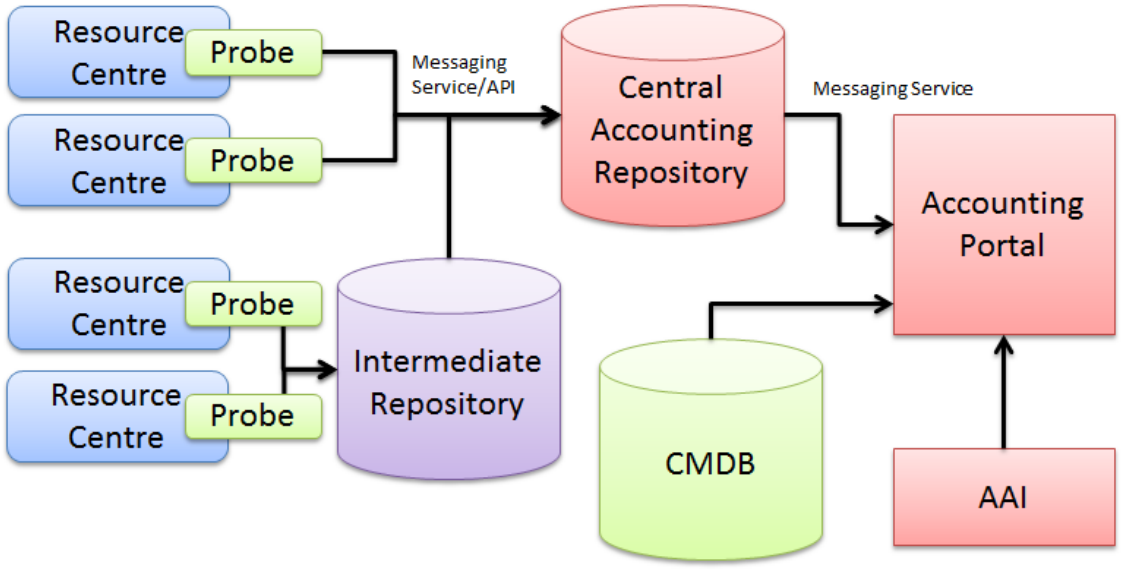


Figure 17. Components of the EOSC Accounting and their interactions

Resource centres that are providing compute or storage to the EOSC infrastructure have to implement a collector (a stand-alone script or program, or a built in function of their resource system) that gathers accounting metrics formatted into a standardized record format (see next section for details). These metrics are then transferred either via a messaging service or by being

retrieved from an API to the Accounting Repository, which stores and processes the data to produce aggregations that are then sent to the Accounting Portal for display.

The Accounting Portal retrieves topology information on how resource centres relate to national infrastructures and regions from a configuration management database (CMDB) and community affiliation from the AAI service to properly organise the accounting data. Information related to groups or VOs should contain also information about scientific disciplines to allow the portal to properly classify the resource usage.

EOSC resource centres can either directly publish accounting information to the EOSC central Accounting Repository or via an intermediate repository that can be related to an infrastructure (European, regional or thematic etc.). It is up to the infrastructure decide to have its own accounting infrastructure connected to the EOSC one or directly leverage the EOSC accounting infrastructure.

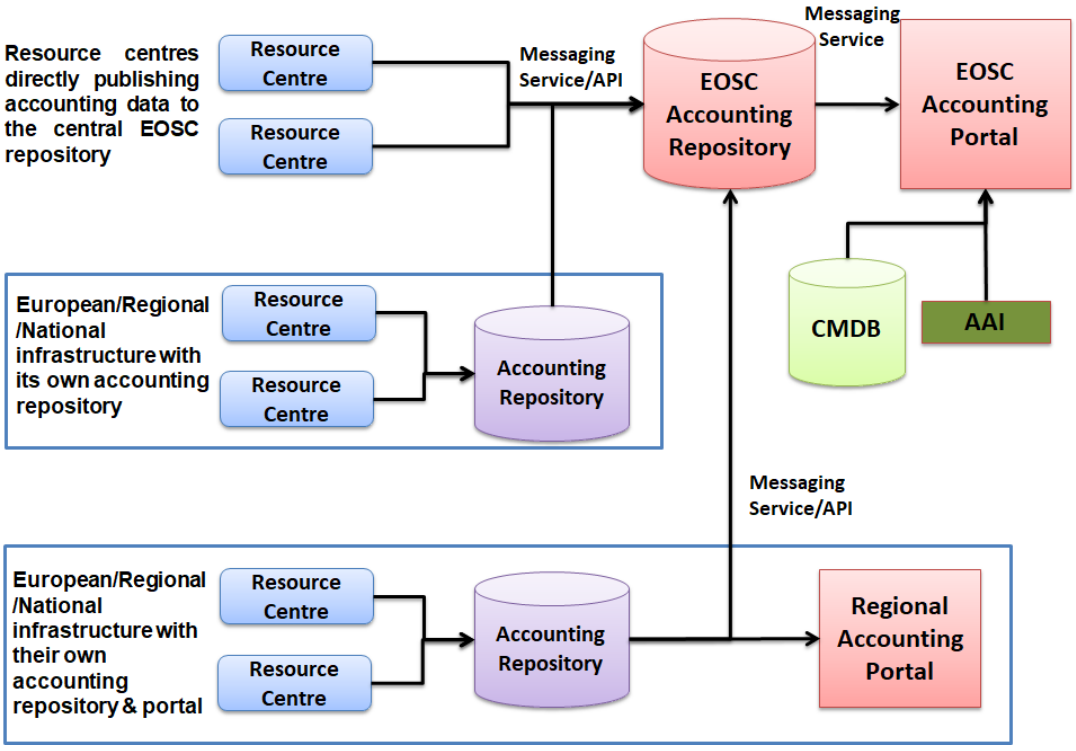


Figure 18. High-level architecture of the EOSC Accounting

Interoperability guidelines

The following interoperability guidelines should be followed to connect an accounting infrastructure to the EOSC accounting infrastructure:

- Standard usage records; to be able to merge accounting data we need to have similar accounting information from the system. The table above lists the standards used.
- Either push to the ARGO Messaging Service (AMS) (https://argoeu.github.io/guides/messaging/) or provide an agreed HTTP API through which accounting data can be gathered.

- Topology information should follow GOCDB guidelines (https://wiki.egi.eu/w/images/d/d3/GOCDB5_Grid_Topology_Information_System.pdf), which allows other infrastructures (e.g OSG[44]) to coexist with separate topologies. The simplest way would be for infrastructures to register in GOCDB[45] or even REBUS[46], the WLCG topology platform. If this is not possible, a GOCDB compatible topology should be provided, with resource centres defined by a definite region, possible subregions and a numeric path identifier that should be consecutive, non-assigned integers, separated by dots (e.g. 1.2.3). An interface to extract the topology information should be provided.
- Metrics and units need to have a compliant format, not only in the datatype, but also the semantics must be commensurable, and the units clear. Before integrating an accounting infrastructure to the EOSC one, the provider of this infrastructure should send to the EOSC Accounting team a list of metrics and related descriptions to be published in the central EOSC Accounting Repository.
- If some of the fields contain URL pointers to metadata, these URLs must be of public access to unprivileged users, at least in a minimal form that can optionally obscure privileged information. In this way meaningful linking from the Portal is allowed.
- AAI should express group membership in a standard way following the EOSC Hub AAI interoperability guidelines (derived from the AARC[47] guidelines).

The EOSC Accounting Repository can accept records produced by any service so long as they are in the correct format and are sent via AMS. Resource providers need to be registered in a configuration management database (e.g. GOCDB) or be individually authorised to publish via AMS.

Examples of solutions implementing this specification

APEL

APEL is an accounting tool that collects accounting data from sites participating in the EOSC infrastructure as well as from sites belonging to other organisations that are collaborating with EOSC. The accounting information is gathered from different collectors into a central accounting repository where it is processed to generate statistical summaries that are available through the EOSC Accounting Portal.

[44] https://opensciencegrid.org/

[45] https://goc.egi.eu/

[46] https://wlcg-rebus.cern.ch/apps/topology/

[47] https://aarc-project.eu/

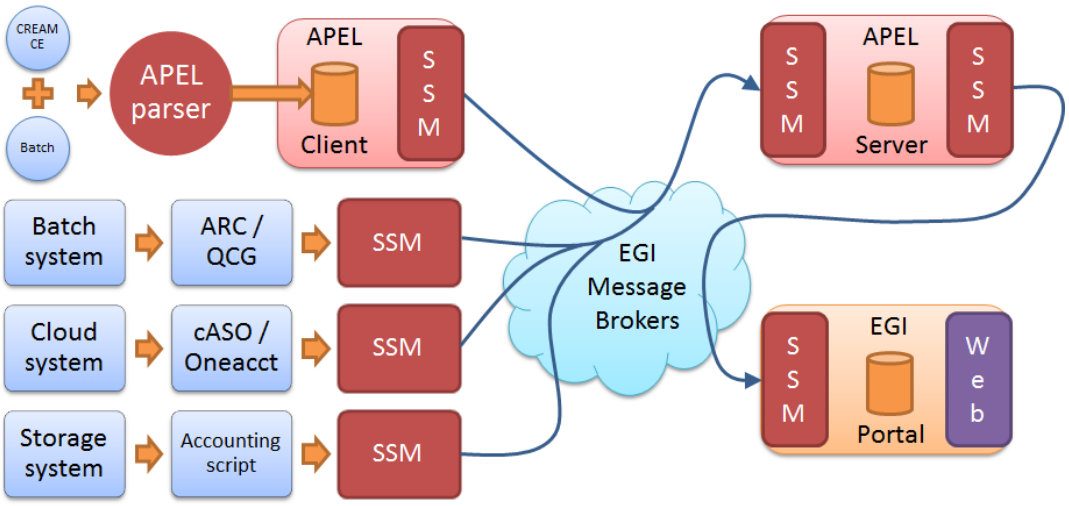


Figure 19. APEL architecture

APEL collects accounting information for compute, cloud and storage resources. Typically a site will deploy some form of accounting collector which will interact with the underlying resource provider and produce an accounting record in a supported format which is then sent to the APEL central repository via the Argo Messaging Service (AMS) and Secure STOMP Messenger (SSM, see https://github.com/apel/ssm). However, APEL is agnostic to the exact source of accounting data, so it is possible to set up regional APEL servers which receive the accounting data from national sites before sending a copy of the information on to the central server.

1. APEL clients (https://github.com/apel/apel) can run an APEL parser to extract data from a batch system and place it in their client database, or they can use third-party tools to extract batch or cloud data. This data is then unloaded into a message format suitable for transmission.
2. APEL clients run a sending SSM to send these messages containing records via the EGI Message Brokers to the central APEL server. The messages can contain either Job Records or Summary records. This is configurable in the APEL client.
3. The central APEL server runs an instance of the SSM, which receives these messages and a "loader" processes the records in the messages and loads them into a MariaDB database.
4. A "summariser" process runs to create summaries of any Job Records received and load them in a "SuperSummaries" table along with any Summary records. This summariser runs as a cron job approximately once a day.
5. A database "unloader" process unloads the summary records into the message format to be sent on by the sending SSM via the EGI Message Brokers to the EGI Accounting Portal.

Accounting Portal

The Accounting Portal receives data from APEL and ultimately from resource centres participating in the EOSC infrastructure as well as from sites belonging to other organisations that are collaborating with EOSC. This is crossed with metadata from other sources to offer an integrated view of accounting data on the Infrastructure.

It is capable of:

- Accounting of CPU time (Normalized or not), Wall Time (Normalized or not), Number of jobs, and efficiency. Grid, Cloud and Storage support
- Grouping of resource usage by Country/Region, date, user group, usage by country.
- Grouping by infrastructure (e.g. WLCG, OSG, etc.)
- Discipline Views
- Views generated by User group Manager, User group Member, Site Admin or User views

DPMT

The Data Project Management Tool (DPMT) used within EUDAT provides an HTTP API that can be used to perform queries to retrieve usage metrics. This is done by the Accounting Repository which then stores the data alongside metrics from other systems. The Accounting Portal then transforms some of the data to enable it to be displayed in aggregated views.

Procedure to integrate a service with the EOSC Hub Accounting

The integration of a resource centre with the EOSC Accounting infrastructure requires two steps:

1. Probes to produce data in the correct format should be installed in the resource centre. The EOSC Accounting Repository will accept records formatting to the standards above. Ready-to-use probes for a large set of resource types are already available (https://github.com/apel/apel/blob/dev/README.md#apel-parsers).
2. Accounting records should be sent to the Accounting Repository. For sending accounting records it is recommended to use SSM to handle the interfacing with AMS, but if it is desired.

8.2.3 Monitoring

Monitoring is the key service needed to gain insights into an infrastructure. It needs to be continuous and on-demand to quickly detect, correlate, and analyze data for a fast reaction to anomalous behavior. The challenge of this type of monitoring is how to quickly identify and correlate problems before they affect end-users and ultimately the productivity of the organization. Management teams can monitor the availability and reliability of the services from a high level view down to individual system metrics and monitor the conformance of multiple SLAs. Monitoring of services, visualization of their status, define availability and reliability reports, dashboard interfacing, sending real-time alerts are some of the key features the Monitoring should support. The dashboard design should enable easy access and visualisation of data for end-users. APIs should also be supported so as to allow third parties to gather monitoring data from the system through them.

Some of the main features of a monitoring system:

- Support of multiple entry points (different types of systems can work together)
- Interoperable
- High availability on the different components of the system
- Support of API's in the full stack so that components are independent in their development cycles
- Support for Multiple Tenants, Configurations, Metrics and profiles to add flexibility and ease of customisation.

Adopted standards

The following table lists the standards recommended in this specification.

Standard	Short description	References
REST	Loosely adhere to the REST paradigm.	[REFERENCE][48]
SAML2	XML based protocol that is used to securely pass the credentials information from Identity provider to Service point (usually web application) that needs it.	[REFERENCE][49]
X509	X.509 is an ITU-T standard for a public key infrastructure (PKI), also known as PKIX (PKI X509)	[REFERENCE][50]
Apache Avro	Data serialization system	[REFERENCE][51]
JSON API	A specification for building apis in JSON format	https://jsonapi.org/

Protocol/API	Short description	References
HTTPS	TLS secured HTTP	REFERENCE[52]
HTTP / JMX / Shell / SQL / Ldap ...	All the plugins should be based on standard protocols or formats	REFERENCE[53]
Nagios Plugin API	Nagios API provide a reference for the monitoring plugin developers.	REFERENCE[54]

[48] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[49] https://wiki.oasis-open.org/security/FrontPage

[50] https://www.rfc-editor.org/info/rfc5280

[51] http://avro.apache.org/

[52] https://tools.ietf.org/html/rfc2818

[53] http://software.in2p3.fr/lavoisier/adaptors.html

[54] https://nagios-plugins.org/doc/guidelines.html

Flink DataStream API	Used to execute live streaming computational jobs on the Flink Streaming platform to produce near real-time results for API & notifications	REFERENCE[55]
Flink DataSet API	Used to execute batch computational jobs on the Flink Streaming platform to produce status and a/r results for Web API	REFERENCE[56]
HDFS API	Used to store ingested monitoring data along with supplementary data (topology, downtimes, weights etc) in distributed HDFS storage.	REFERENCE[57]
ARGO API over REST API	The ARGO Web API provides the Serving Layer of ARGO. It is comprised of a high performance and scalable data store and a multi-tenant REST HTTP API, which is used for retrieving the Status, Availability and Reliability reports and the actual raw metric results.	REFERENCE[58]

High-level Service Architecture

The service collects status (metrics) results from one or more monitoring box(es) and delivers daily and/or monthly availability (A) and reliability (R) results of distributed services. Both status results and A/R metrics are delivered through a Web UI, with the ability for a user to drill-down from the availability of a site to individual test results that contributed to the computed figure.

[55] https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/datastream_api.html

[56] https://ci.apache.org/projects/flink/flink-docs-release-1.8/dev/batch/

[57] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[58] http://argoeu.github.io/guides/api/

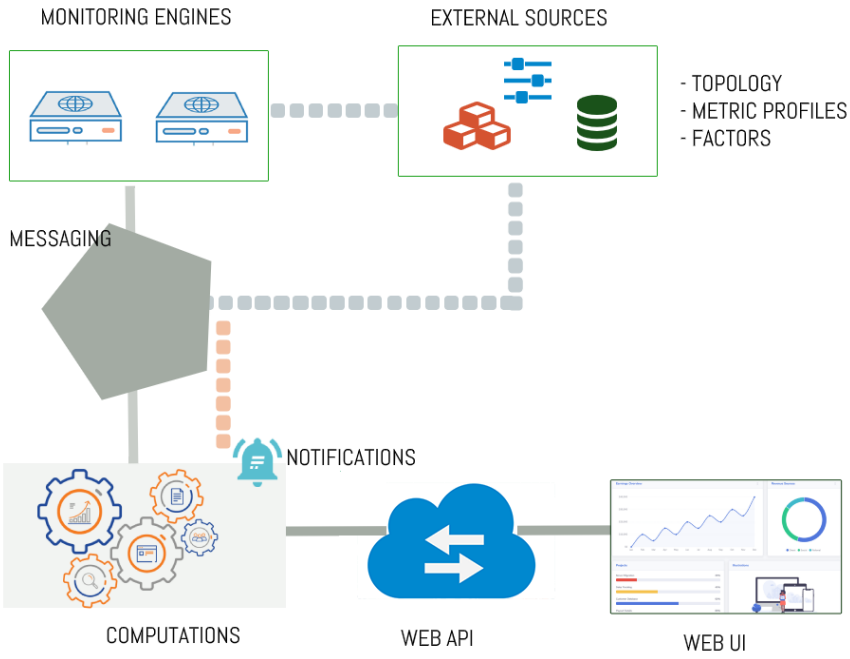


Figure 20. High level architecture of a Monitoring service

The main components of a monitoring service are depicted in the high-level architecture diagram and described below.

Monitoring Engine: This service component executes the service checks against the infrastructure and delivers the metric data (probe check results) to the Messaging Service.
Metric and Profile Management Component: This service component is used in order to define checks (probes) and associate them to service types. Each grouping of checks and service types forms a profile.

Computations & Analytics: This component of the system should include computational job definitions for ingesting data, calculating status and availability/reliability and a management service to automatically configure, deploy and execute those jobs on a distributed processing engine for stateful computations. At the same time this component analyzes the monitoring results and send notification based on a set of rules, to inform the users (operators, NGIs) about the status of their services.
The result of the computations should be stored in a distributed file system (in a highly fault-tolerant system). It should provide high throughput access to application data and is suitable for applications that have large data sets. Apart from the storage of the raw data in a distributed file system, data should also be stored in a document database designed for ease of development and scaling.

WEB API: Rest-like HTTP API service that provides access to status and availability/reliability results. It supports token-based authentication and authorization with established roles. Results are provided in JSON Format.

WEB UI: The Web UI is the component used to store, consolidate and "feed" data into the web application. The global information from the primary and heterogeneous data sources retrieved by means of the use of the different plugins. The collected information is structured and organized within configuration files in the service and, finally, made available to the web application without the need for any further computations.

The resulting data is exposed as XML views through a RESTful web service interface .

This modular architecture is conceived to add easily new data source in this model and use the cached information if a primary source is unavailable. With the help of this component it is quite easy to add a new source of information. So this component ensures the interoperability of the portal with other tools/services. If the service to be integrated uses standard protocols or formats through the use of existing plugins provided.

Interoperability guidelines

This section presents the main integration and usage use cases for monitoring in EOSC and proposes ARGO[59] interfaces as guidelines to be followed to achieve the interoperability between monitoring systems in EOSC.

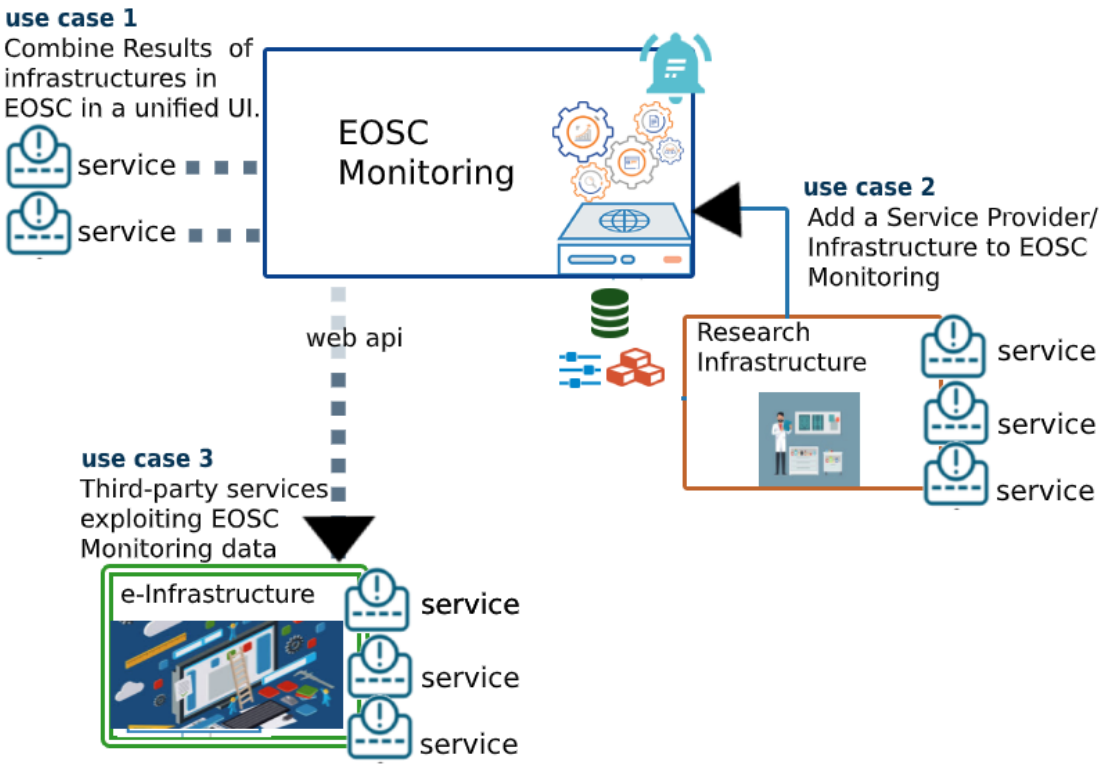


Figure 21. Integration scenarios in the EOSC Monitoring

[59] https://argoeu.github.io/

Use Case 1: Combine Results of one or more infrastructures in EOSC in a unified UI.

The proposed system should be able to combine the A/R results from different providers / infrastructures in a unified web view. The general goal of "distributed monitoring" is to allow different infrastructures with the same or different environment to scale. There are a number of different options for supporting this. All of them are based on the concept that different sources will publish status and performance data in a predefined form that is read/scrapped from the core engine of the monitoring service. The data should also be stamped with their source and timestamp. Every metric should be prefixed with [source_type], following the metric naming best practices (based on nagios). Every metric is also labeled with the hostname and service description. These predefined messages should be sent to the Messaging system which is the service responsible to pass them to the computations engine which performs the necessary calculations to produce the reports.

How Argo Solves this
Argo Compute engine uses as source the results of the probes sent by two or more monitoring engines (nagios boxes) via the Argo Messaging Service. Metric data comes in the form of avro files and contains timestamped status information about the hostname, service and specific checks (metrics) that are being monitored. A typical item of information in the metric data avro file contains the field listed in the table below. The compute engine calculates the Availability and Reliability of each service group based on the instructions and mapping given by the Topology and Metric Aggregation & Threshold profiles[60]. This fact allows the compute engine to be flexible enough in order to combine results from a number of sources and produce reports for almost any combination possible it is therefore able to produce integrated views that combine the topologies of more than one Service Provider or Infrastructure Providers.

Name	Description	Required
hostname	The fqdn address of the host being monitored	YES
service	The name of the specific service being monitored	YES
metric	The name of the specific metric (check) of the service that is being monitored	YES
timestamp	Time of the monitoring check	YES

[60] http://argoeu.github.io/guides/argo-compute-engine/input/

status	Status of the metric during the monitoring check	YES
monitoring_host	The fqdn of the monitoring agent	NO
summary	Text containing a summary of the monitoring check	NO
message	Text containing the detailed system output message of the monitoring check probe	NO
tags	Array containing optional user defined tags	

Use Case 2: Add a Service Provider/Infrastructure to EOSC Monitoring

In order to add support for a new Service Provider or Infrastructure in EOSC Monitoring Service, the provider should only need to provide the topology of the services to be monitored and the equivalent metric and aggregation profiles. The system should take care of all the actions required to probe every endpoint in the topology with the metrics/probes defined and aggregate the results according to the profiles defined and present them in a Web-UI.

How Argo Solves this
For each new Tenant ARGO uses as topology input the xml feed by EOSC CMBD[61]s and provides POEM[62], a component to allow management of the necessary profiles that are required to configure monitoring engines automatically in order to run the necessary probes. The results are then passed through the Argo Messaging Service to the Argo Compute Engine[63] which performs the necessary calculations to produce the Availability and Reliability of each endpoint, service and service group defined in the topology according to the Aggregation profiles and present then serve the results via the ARGO-Web-API[64] to be rendered/presented by the WEB-UI[65].

Use Case 3: Third-party services exploiting EOSC Monitoring data

Any EOSC service should be able to retrieve and use the status information and metrics computed by the EOSC Monitoring system. An API should be provided to allow any authorised third-party service to retrieve such data.

[61] EOSC Configuration Management Databases GOCDB (goc.egi.eu) and DPMT (dp.eudat.eu)
[62] http://argoeu.github.io/guides/poem/
[63] http://argoeu.github.io/guides/argo-compute-engine/
[64] http://argoeu.github.io/guides/api/
[65] http://argoeu.github.io/guides/webui/

How Argo Solves this
The ARGO Web API[66] comprises a high performance and scalable data store and a multi-tenant REST HTTP API, which is used for retrieving the Status, Availability and Reliability reports and the actual raw metric results.

Examples of solutions implementing this specification

ARGO

ARGO is a flexible and scalable framework for monitoring status, availability and reliability of services provided by infrastructures with medium to high complexity. It can generate multiple reports using customer defined profiles (e.g. for SLA management, operations etc.) and has built-in multi-tenant support in the core framework.

ARGO supports flexible deployment models and its modular design enables ARGO to integrate with external systems (such as CMDBs, Service Catalogs etc.). During the report generation, ARGO can take into account custom factors such as the importance of a specific service endpoint, scheduled or unscheduled downtimes etc.

For the Availability & Reliability monitoring, ARGO relies on a modular architecture comprised of several components described in the next subsections.

The ARGO Monitoring Engine

For status monitoring, ARGO relies on Nagios. All probes developed for ARGO follow the Nagios conventions and can run on any stock Nagios box. ARGO provides an optional set of addons for the stock Nagios that provide features such as auto-configuration from external information sources, publishing results to external Message Brokers etc.

In order to use the new messaging service, the monitoring engine also supports the new AMS Publisher. The AMS publisher is a new component acting as a bridge from Nagios to ARGO Messaging system. It is an integral part of software stack running on ARGO monitoring instance and is responsible for forming and dispatching messages that are results of Nagios tests. Ready and running on devel infrastructure. It is running as a UNIX daemon and it consists of two subsystems:

- queueing mechanism
- publishing/dispatching part

Messages are cached in local queue with the help of OCSP Nagios calls and each queue is being monitored by the daemon. After configurable number of accumulated messages, publisher that is associated to queue sends them to ARGO Messaging system and drains the queue. argo-nagios-ams-publisher is written in multiprocessing manner so there is support for multiple queue/publish pairs where for each, new worker process will be spawned.

The ARGO Connectors

[66] http://argoeu.github.io/guides/api/

Through the use of custom connectors, ARGO can connect to multiple external Configuration Management Databases and Service Catalogs. Already there are connectors for the EGI and EUDAT e-Infrastructures.

The ARGO Consumer

The ARGO Consumer is ingesting monitoring results in real-time from external Message Brokers. The consumer is responsible for the initial pre-filtering of the monitoring results and encodes them using AVRO serialization format before passing to the Compute Engine.

The ARGO Compute Engine

A powerful and scalable analytics engine built on top of Hadoop and HDFS. The Compute Engine is responsible for the aggregation of the status results and the computation of availability and reliability of composite services using customer defined algorithms. The reorganization of the Compute Engine to support stream processing in real time is one of the key new factors. A new streaming layer is introduced. Monitoring results flow through the AMS, to the streaming layer (in parallel to the HDFS). The streaming layer is used in order to push raw metric results to the metric result store and to compute status results and push them to the status store in real-time.

The ARGO Web API

The ARGO Web API provides the Serving Layer of ARGO. It is comprised of a high performance and scalable datastore and a multi-tenant REST HTTP API, which is used for retrieving the Status, Availability and Reliability reports and the actual raw metric results.

The ARGO Web UI

The default web UI is based on the Lavoisier Data Aggregation Framework[67].

ARGO has been adopted by

- EGI infrastructure
- EUDAT infrastructure

Procedure to integrate a service with the EOSC Hub Monitoring

Follow the steps

1. GGUS ticket at ARGO/SAM EGI Support Unit with:
    a. Small description of the integration - use of the service
    b. A name for the new project - infrastructure / project / service to monitor

2. The Monitoring team will create a new project into the development infrastructure for testing.

3. If the request refers to a new service type / probe, then the probe should follow the guidelines mentioned in the interoperability section[68].

[67] http://software.in2p3.fr/lavoisier/

[68] https://wiki.eosc-hub.eu/display/EOSC/ARGO+Guidelines+for+monitoring+probes

8.2.4 Software Quality Assurance

The Software Quality Assurance (SQA) is the process responsible for the overall supervision of the software development lifecycle ensuring that the required quality level is achieved. The SQA encompasses all software development processes starting from the definition of requirements, coding, release, testing and integration.

This technical area covers ways to deliver quality software for EOSC consumption and favours the adoption of automated solutions over the traditional manual-based validation mechanisms. The automation allows not only to speed up the development tasks but as well improves the reliability of the developments "ensuring the fast execution of defined tests at each change in the codebase" and keeping them aligned with the initial user requirements and design "Fast feedback received at any development stage - faster release of quality software".

Adopted standards

The present document follows the well-known practices and standards adopted by the open-source community.

Standard	Short description	References
IEEE 730-2014	This standard establishes the requirements for initiating, planning, controlling, and executing the Software Quality Assurance processes of software development.	https://standards.ieee.org/standard/730-2014.html
ISO/IEC/IEEE 12207:2017	International Standard - Systems and software engineering -- Software life cycle processes - establishing a common framework for software life cycle processes	https://standards.ieee.org/standard/12207-2017.html

Guidelines	Short description	References
Common SQA Baseline Criteria for Research Projects	A set of Common Software Quality Assurance Baseline Criteria for Research Projects	White paper for SQA Baseline for Research projects v3.0
EGI QC 7	7th release of the EGI Quality Criteria, that is used for the validation of software products within EGI's Software Provisioning Process	http://egi-qc.github.io/
Semantic Versioning	Best practices for handling software	https://semver.org/

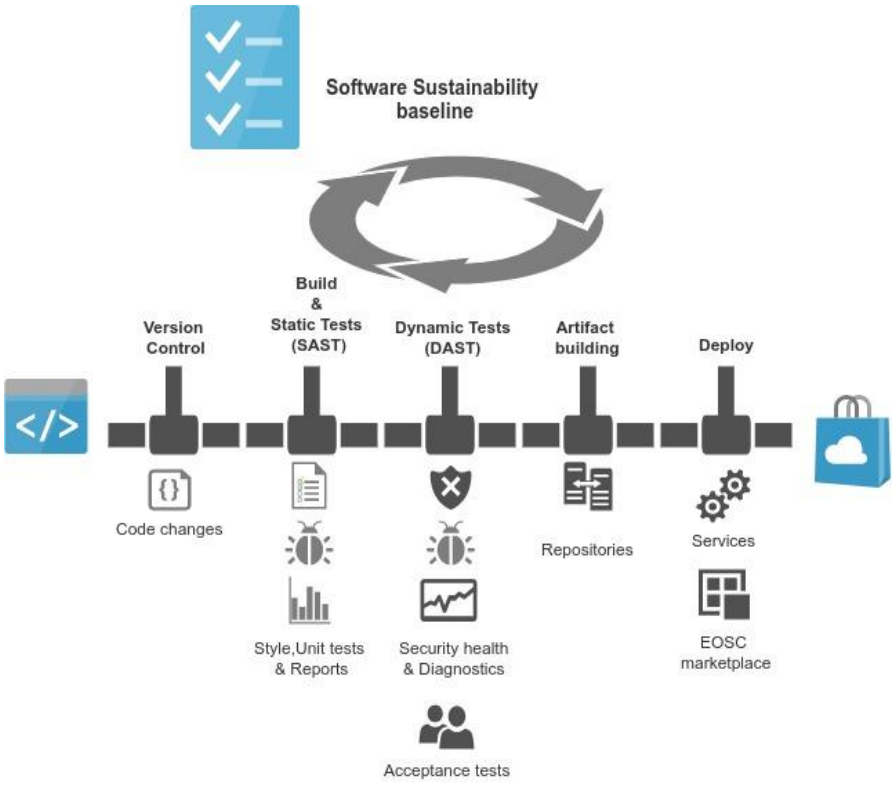
	versions	

High-level Service Architecture

The SQA delivers assistance to service providers throughout the software development lifecycle in order to attain desired quality and timely delivery in the software produced, and consequently, promote the maturity of the EOSC services.

Even if currently, there are no specific EOSC-hub services that offer support for the assessment of the software quality requirements range from code review to static and dynamic testing, which includes both security and interoperability tests. The assessment of compliance with the quality requirements is implemented using continuous integration (CI) and delivery (CD) pipelines, where successfully produced artefacts can be made readily available through the EOSC repositories.

A possible architecture for the Integration of sustainable software for quality services into EOSC is depicted in the following figure:



Interoperability guidelines

The following items make a short description of the topics relevant to EOSC:

Code Accessibility:

Following the open source model, the source code shall be open and publicly available through social coding platforms in order to increase its visibility and foster collaboration.

Licensing:

Licenses must be physically present (e.g. as a LICENSE file) in the root of all the source code repositories related to the software component.

Versioning:

Semantic versioning is recommended for tagging the new software releases, avoiding any dependency conflict.

Documentation:

Meaningful and differentiated documentation must be available and maintained for each specific audience of the given software: user, admin, and developer. It shall be available online (using a documentation repository) and preferably produced using a markup language (such as Markdown or reStructuredText). Thus, the documentation is treated as code using a VCS.

Code Workflow:

Software is best managed by means of a version control system (VCS) solution, which facilitates the adoption of a branching model to conduct the development. Thus, the production version of the software remains in a working state, while the new features or bugs are added. Moreover, several versions of the software can be maintained simultaneously, such as long-term support (LTS) versions.

Code metadata:

Software shall be uniquely identified via a persistent identifier so that it can be easily discovered, reused, citable and preserved. Adding metadata to describe the software (in the code repository) is the first step towards its identification.

Security:

Security assessment shall be continuously performed on every change in the source code. Tools for security static (SAST) and dynamic analysis testing (DAST) already cover the most common security flaws in the code and in the running services.

Code review:

Human oversight of the changes done in the code shall be the last step in the assessment of each new feature or bug implemented, once the test phase has been completed. The suitability of the changes implemented, the statements and/or libraries used, and the security review tasks are commonly associated with code reviews.

Examples of solutions implementing this specification

Some software products delivered under EOSC are already compliant with the quality conventions described in the Interoperability Guidelines section. Examples of such software products and their respective continuous integration and delivery pipelines are:

- Infrastructure Manager
  - https://github.com/indigo-dc/im/blob/master/Jenkinsfile
- udocker
  - https://github.com/indigo-dc/udocker/blob/master/Jenkinsfile
- PaaS orchestrator
  - https://github.com/indigo-dc/orchestrator/blob/master/Jenkinsfile
- cloud-info-provider
  - https://github.com/EGI-Foundation/cloud-info-provider/blob/master/Jenkinsfile
- WaTTs
  - https://github.com/indigo-dc/wattson/blob/master/Jenkinsfile
- oidc-agent
  - https://github.com/indigo-dc/oidc-agent/blob/master/Jenkinsfile

Procedure to integrate a service with the EOSC Hub Monitoring

Currently, there are no specific EOSC-hub services that offer support for the assessment of the software quality nevertheless the SQA already provides some ready-to-use continuous integration and delivery pipelines based on automation services such as Jenkins. To this end, a library with the most common functionalities needed during the testing and delivery phases is available.

So, a service provider must fulfil the following conditions:

1. Source code must reside on a hosting service repository with version control (e.g. GitHub).
2. Licensing, Documentation and versioning must be publicly accessible.

In addition, service providers should:

3. Deploy an automation service (e.g Jenkins) and adapt pipelines[69] in order to make automatic tests. A library for Jenkins with some common functionalities needed to implement the testing and delivery phases is already provided[70].

[69] https://jenkins.io/doc/book/pipeline/

[70] https://github.com/indigo-dc/jenkins-pipeline-library

8.3 Security

8.3.1 Security Incident Response Trust Framework for Federated Identity (SIRTFI)

As a TCOM area, this specification describes Security, i.e. the standards and specifications for operational security, or "cybersecurity." Ultimately, the purpose of security, in this sense, is to ensure that the infrastructure is trustworthy, and participants are able to carry out their legitimate work and collaborations, while protecting the infrastructure and data from unauthorised parties.

In order to ensure that participants in e-infrastructures, research infrastructures, and identity federations (such as those operated by NRENs) can reduce the risk of security incidents, and collaborate on investigating, managing, and resolving security incidents, it is necessary to have a shared security operations framework. Specifically, this will cover:

- best practices,
- security contacts,
- processes for assessing severity (and hence urgency),
- traceability of users,
- defining, updating, and tracking users' acceptance of acceptable use policies.

In addition, the standards cover how the compliance is asserted in a machine-readable way. There are also constraints on human readable information but the specification on how to implement these constraints is left to the federation operator and/or participants.

It should also be noted that the wider issue of establishing, maintaining, and restoring trust - between organisations, communities, and infrastructures - is not covered here.

Adopted standards

The standards listed below are formally issued by REFEDS (Research and Education FEDerationS) and IGTF (Interoperable Global Trust Federation), respectively. However, both have come out of AARC2 NA3 work (= policies and harmonisation), and are established on the basis of wide consultation, not just in Europe.

Standard	Short Description	References
Security Incident Response Trust Framework for Federated Identity (SIRTFI)	Best practices for ensuring that federation participants are capable of minimising the risk of security incidents and collaborate on handling them. The standard applies to both organisations running IdPs and SPs.	https://refeds.org/sirtfi
Scalable Negotiator for a Community Trust Framework in Federated	Practices for handling and communicating SIRTFI compliance of federation participants in proxy-based federations.	https://www.igtf.net/snctfi/

Infrastructures (SNCTFI) -		
A Trust Framework for Security Collaboration among Infrastructures	Operational security requirements on the infrastructure as a whole, published by the WISE community [7]. Overlap with SIRTFI (which covers IdPs and SPs).	https://wise-community.org/wp-content/uploads/2017/05/WISE-SCI-V2.0.pdf

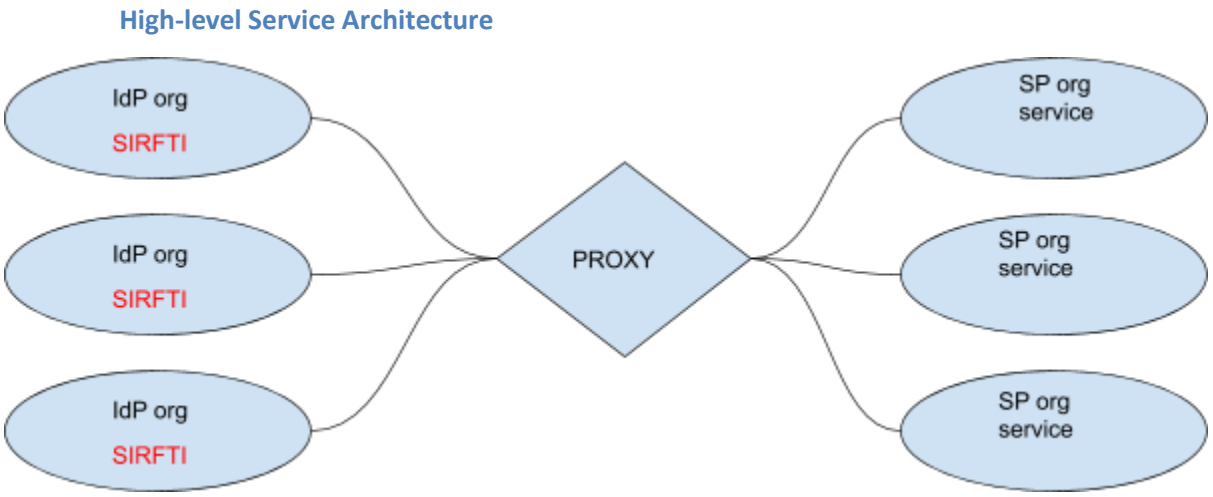


Figure 22: High level architecture diagram of SIRTFI

Interoperability guidelines

The standards specify how SIRFTI compliance should be asserted in SAML-based federation (in the metadata). SNCTFI is specified to enable proxy-based federations[71] to communicate the relevant attributes (SIRTFI compliance, traceable user identities) in a trustworthy way across proxies.

In addition, there is guidance on activities and practices that are relevant to the implementation of SIRTFI and SNCTFI. Guidance on a specific topic may be published by different projects or organisations - sometimes by national cybersecurity organisations - and should not vary substantially, although some might be more thorough than others. Although these are technically not standards, most of the guidance listed here is, like standards, based on state of the art and wide consultations.

In our guidance table, we have endeavoured to find examples of guidance likely to be accepted across a wide range of infrastructures.

Guideline	Short Description	Reference

[71] Authentication and Authorisation for Research Communities (AARC) https://aarc-project.eu/

Computer Security Incident Handling Guide	Principally focuses on handling a single incident but also includes sharing information with a Computer Emergency Readiness Team (CERT)	NIST SP800-61 rev 2 DOI:10.6028/NIST.SP.800-61r2
Common Vulnerabilities and Exposures	The current list of known vulnerabilities can help organisations prevent incidents	https://cve.mitre.org/

Most countries would have national cybersecurity organisations. Organisations would also have their own policies and processes. There is also cybersecurity professional organisations, both nationally and internationally[72]. An example of the latter is (ISC)[2], which publishes a code of ethics of cybersecurity professionals, as well as a certification scheme, CISSP. Also, ENISA has cybersecurity training[73].

It should be added that there are many commercial "solutions" for (usually organisational) cybersecurity. The state of the art comprises:

- Cybersecurity awareness training for employees;
- Ransomware protection;
- Endpoint protection and security testing; penetration testing (pentesting);
- Assistance with security incident handling from mitigation (phishing exercises, code analysis), through forensics to reactive (intrusion detection, SIEM, etc.) and to proactive handling (e.g. threat hunting);
- Virtual Private Networks for access to corporate resources;
- Tools to detect unusual or suspicious activities, e.g. login from an unusual location which might require multi-factor authentication, or detection of insider threats ("compromised" employees who access data they shouldn't).

Note that a security evaluation should include a threat model which should also cover any additional resources used by the community. These can include, but are not limited to, connecting users to infrastructures with mobile phones (e.g. for second factor authentication), community-specific edge devices such as sensor networks that provide data to the community's research infrastructure, and external clouds used by the community.

Examples of solutions implementing this specification

EGI

EGI references guidance on SIRTFI to its IdPs[74].

[72] https://www.enisa.europa.eu/topics/national-cyber-security-strategies

[73] https://www.enisa.europa.eu/topics/trainings-for-cybersecurity-specialists

[74] https://wiki.egi.eu/wiki/AAI_guide_for_IdPs

Notably, EGI also runs a Security Vulnerability assessment Group (SVG[75], which handles the vulnerabilities related to software. Led by Dr Linda Cornwall from UKRI-STFC, the group is currently (Jan. 2020) in the process of establishing a deployment vulnerability group for EOSC.

EUDAT

During the lifetime of the EUDAT2 project, the project's WP6 specified that participants should adhere to SIRTFI (the reference does not seem to be publicly available). In particular, the project maintained a link of security contacts for each organisation, although there was an issue with keeping the page up to date.

GEANT

From Terena/GEANT, it is worth noting:

- TF-CSIRT[76] working group
- The Information Security Management Special Interest Group (SIG-ISM[77])
- The WISE[78] community which includes SCI[79] which published.
- The CSIRT-KIT[80] project

ENISA

The European Union Agency for Cybersecurity provides guidance[81] on incident reporting, and extensive guidance on operating CSIRT[82] services, and a lot of other relevant information on cybersecurity.

NRENs

Currently NRENs do not require SIRTFI for their participants, but they support it for organisations that wish to assert it.

It was noted that when CERN's eduGain authentication started rejecting IdPs that did not assert SIRTFI, the uptake of SIRTFI improved.

[75] https://wiki.egi.eu/wiki/SVG

[76] https://wiki.geant.org/display/TTC/Report+on+TF-CSIRT+Membership

[77] https://wiki.geant.org/display/SIGISM/SIG-ISM+Home

[78] https://wise-community.org/

[79] https://wise-community.org/wp-content/uploads/2017/05/WISE-SCI-V2.0.pdf

[80] http://www.csirt-kit.org/

[81] https://www.enisa.europa.eu/topics/incident-reporting

[82] https://www.enisa.europa.eu/topics/csirt-cert-services

9 Technical Specifications for Common services

9.1 Data Publishing and Open Data

9.1.1 Digital Repository

A digital repository is an infrastructure component that is able to store, manage and curate Digital Objects and return their bitstreams when a request is being issued. A digital object (DO) is represented by a bitstream, is referenced and identified by a persistent identifier and has properties that are described by metadata. Digital Objects can be aggregated to digital collections. A Digital Collection is in principle a complex Digital Object which is again identified by a PID and described by metadata. Metadata contains descriptive, contextual and provenance assertions about the properties of a DO and/or DC.

This description is based on terms[83] defined by the RDA Data Foundation and Terminology Working Group.

High-level Service Architecture

In Figure 18 you can find a high-level diagram of a digital repository. In the diagram the macro features of a digital repository have been indicated.

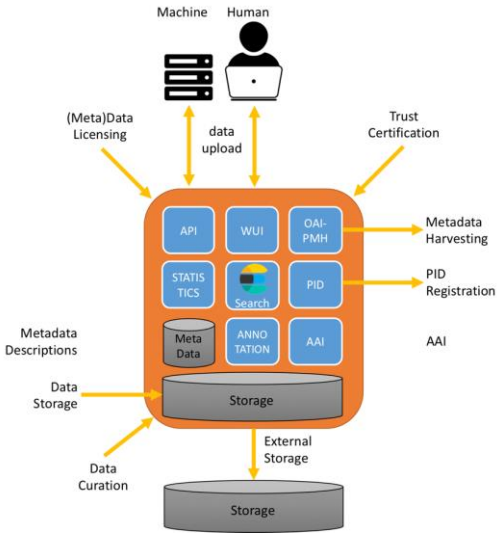


Figure 23: High level architecture diagram of a Digital Repository

Macro Features

Protocols for up-/download and/or to publish of digital objects

To upload/download or to manage digital objects or collections of digital objects in a data publishing platform a different, a richer protocol is required then a basic data transfer protocol. This is mostly

[83] http://hdl.handle.net/11304/5d760a3e-991d-11e5-9bb4-2b0aad496318

due to the fact that digital objects are more complex, contains more information which needs to be handled as a basic bitstream

Storing data and preserving bitstreams

A basic feature of a digital repository is to store and preserve bitstreams of digital objects and digital collections over time. Therefore, the infrastructure on which the bitstreams are stored should be based on a reliable storage infrastructure in which multiple copies of the bitstreams are stored, integrity of the data is ensured via checksums and regular checks are run to prevent against bit rot.

De storage infrastructure on which the bitstreams are stored can be implemented in many different ways and strongly depends on local choices of the digital repository infrastructure provider, the technology choice and specific requirements to support the digital repository owner use cases, for example on data volume, number of objects, data organisation. Frequently use storage infrastructures are RAID storage arrays with tape backups to ensure reliability, hierarchical storage infrastructures in which data is automatically migrated between different storage layers and in which multiple copies are maintained, or in an object store in which multiple copies are ensured.

To prevent data loss, digital repository owners should think about storing bitstreams at physical separated locations. It is difficult to give a minimum distance between the two physical locations; this strongly depends environmental conditions of the different locations.

Metadata descriptions

Metadata contains descriptive, contextual and provenance assertions about the properties of a Digital Object or a Collection of Digital Objects. Many communities have defined their own community-specific metadata schema's. To support the exchange of metadata for publishing and harvesting different minimum metadata standards have been defined.

Metadata harvesting

To enlarge the discoverable and findability of scientific artificers, scientific repositories should enable the harvesting of metadata.

Persistent Identifier

A persistent identifier is a long-lasting ID represented by a string that uniquely identifies a DO and that is intended to be persistently resolved to meaningful state information about the identified DO. Different types persistent identifiers for different purposes are available. In this context, persistent identifiers for the purpose of publishing or to refer to digital objects are depicted.

License

A license is a legal instrument that describes the conditions under which the data can be used and reused. Different type of licenses can be applied on different type scientific artefact, for example for publications, research data and/or software.

Search

The search functionality provides an easy way to find and filter digital collections and objects on basis of certain search criteria. The search criteria are commonly based on specific keywords of the metadata, words within the metadata description, types of digital objects, etc. The search

functionality provided strongly depends on the technology which is used to build the digital repository. Frequently the search functionality is based on SOLR.

Authentication and Authorisation

In general data owners of the digital collections and objects stored and made available through a digital repository are registered users of the digital repository and have therefore granted access. Before a data owner can upload data to the digital repository, he/she must authenticate themselves. The registration and authentication users involve personal information, the digital repository and digital repository service provider must comply to the Rules for the protection of personal data inside and outside of the EU (GDPR)

Data Curation

According to the University of Illinois' Graduate School of Library and Information Science, Data Curation is defined as the active and on-going management of data through its lifecycle of interest and usefulness to scholarship, science, and education; curation activities enable data discovery and retrieval, maintain quality, add value, and provide for re-use over time.

Data Curation can be defined at different levels (e.g. storage media, bitwise, data format, content and context) and within the different phases of the data life, at creation level, during processing time and over time while preserving data long term. To be able to comply to the Trust certification schemes it is required to define and describe the minimum level curation applied within the digital repository.

Trust Certifications

To show and to provide trust to users and to researchers who want to publish and want to share their research data to the public (i.e. open access) digital repository providers can certify the digital repository according to one of the certification frameworks specific for digital repositories.

Adopted Standards

Below a list of common standards, protocols and API's frequently adopted by data publishing standards.

Protocols for up-/download and/or to publish of digital objects

Protocol/API	Short description	References
SWORD	SWORD is a lightweight protocol for depositing content from one location to another. It stands for Simple Web-service Offering Repository Deposit and is a profile of the Atom Publishing Protocol. SWORD is supported by a number of digital repository technologies	http://swordapp.org/

DOIP	The Digital Object Interface Protocol is a core protocol of the Digital Object Architecture (DO Architecture; or DOA). The DO Architecture is a logical extension of the Internet architecture that addresses the need to support information management more generally than just conveying information in digital form from one location in the Internet to another.	https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf
FedoraCommons	Example of a technology specific API	https://wiki.duraspace.org/display/FEDORA475/RESTful+HTTP+API
DSpace	Example of a technology specific API	https://wiki.duraspace.org/display/DSDOC6x/REST+API
B2SHARE	Example of a service specific API	https://b2share.eudat.eu/help/api
Zenodo	Example of a service specific API	http://developers.zenodo.org/#rest-api

Metadata guidelines for metadata harvesting

Minimum metadata guidelines to support metadata harvesting by generic research related metadata aggregator and search engines.

Standard	Short description	References
DataCite	The DataCite Metadata Schema is a list of core metadata properties chosen for an accurate and consistent identification of a resource for citation and retrieval purposes, along with recommended use instructions.	https://schema.datacite.org/
OpenAIRE guidelines for Data Archives	The OpenAIRE Guidelines for Data Archive Managers 2.0 will provide instruction for data archive managers to expose their metadata in a way	https://guidelines.openaire.eu/en/latest/data/index.html

	that is compatible with the OpenAIRE infrastructure. This is a platform specific guideline.	
B2FIND Guidelines	The EUDAT's guidelines for the metadata service B2FIND for data providers. These guidelines are intended to provide information about the requirements for successful integration in B2FIND. This is a service specific guideline.	http://b2find.eudat.eu/guidelines/index.html
EDMI	A minimum information metadata guideline to help users and services to find and access datasets reusing existing data models and interfaces. EDMI has been defined in the EOSCpilot project.	https://eosc-edmi.github.io/
Schema.org	Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.	http://schema.org/

Community specific guidelines for metadata

Many communities have defined and maintain metadata standards to be used within their own community and/or science domain. In the table below you can find a non-exhaustive list of references to some example communities.

Communities	Science Domain	Short description	References

ELIXIR, BBMRI	Life Sciences	Bioschemas aims to improve the Findability of data in the life sciences. It does this by encouraging people in the life sciences to use Schema.org markup in their websites so that they are indexable by search engines and other services.	https://bioschemas.org/
CLARIN	Linguistics	The CLARIN Standards Information System lists language-technology-related standards that CLARIN centres are willing to accept and recognize and visualizes some of their interdependencies.	https://clarin.ids-mannheim.de/standards/
ENES-IS	Climate	Data and metadata standards used within the ENES Climate community.	https://portal.enes.org/data/data-metadata-service/standards
Europeana	Cultural Heritage	The Europeana Data Model (EDM) is a new proposal for structuring the data that Europeana will be ingesting, managing and publishing.	https://pro.europeana.eu/resources/standardization-tools/edm-documentation

Under auspices of the Research Data Alliance a list of metadata standards in use by communities is maintained.

Metadata harvesting

Protocol/API	Short description	References
OAI-PMH	The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. Data Providers are repositories that expose structured metadata via OAI-PMH. Service Providers then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP.	https://www.openarchives.org/pmh/
ResourceSync	This ResourceSync specification describes a synchronization framework for the web	http://www.openarchives.org/rs/toc

	consisting of various capabilities that allow third-party systems to remain synchronized with a server's evolving resources.	
OGC/CSW	Domain specific standard from the Open Geospatial Consortium. OGC Catalogue interface standards specify the interfaces, bindings, and a framework for defining application profiles required to publish and access digital catalogues of metadata for geospatial data, services, and related resource information.	https://www.opengeospatial.org/standards/cat

Persistent Identifier

In the table below you can find a non-exhaustive list of references to some example persistent identifier types in use.

Standard	Short description	References
DOI	A DOI name is permanently assigned to an object to provide a resolvable persistent network link to current information about that object, including where the object, or information about it, can be found on the Internet. While information about an object can change over time, its DOI name will not change.	https://www.doi.org/
EPIC	EPIC PIDs are based on the Handle system and can be used to provide persistent references to digital objects.	https://www.pidconsortium.eu/
ARK	ARKs are URLs designed to support long-term access to information objects, such as digital objects, physical objects, living beings and groups and/or intangible objects.	https://n2t.net/e/ark_ids.html
PURL	PURLs are Web addresses or Uniform Resource Locators (URLs) that act as permanent identifiers in the face of a dynamic and changing Web infrastructure.	http://www.purlz.org/

Identifiers.org	The Identifiers.org Central Registry service provides a centralized directory of Compact Identifiers. The service is part of the ELIXIR infrastructure.	http://identifiers.org/

License

Purpose	Short description	References
Research data	Creative Commons licenses provide an easy way to manage the copyright terms that attach automatically to all creative material under copyright. Creative Commons offers a core suite of six copyright licenses.	https://creativecommons.org/licenses/
Source code	Creative Common licenses are not recommended for source code, because they do not contain specific terms for distributing source code. For this purpose, specific licenses are available for distributing source code, for example GPL, Apache, BSD, MIT.	https://opensource.org/licenses

Trust Certifications

Standard	Short description	References
CoreTrustSeal	The CoreTrustSeal is the basic certification level for trusted digital repositories. Within the CoreTrustSeal service providers operating a digital repository must comply with the 16 requirements of the CoreTrustSeal.	https://www.coretrustseal.org/
DIN 31644	DIN 31644 is an extended certification procedure with higher requirements compared to CoreTrustSeal.	https://www.din.de/de/mitwirken/normenausschuesse/nid/normen/wdc-beuth:din21:147058907
ISO 16363	ISO 16363:2012 defines a recommended practice for assessing the trustworthiness of digital repositories. It is applicable to the entire range of digital repositories.	https://www.iso.org/standard/56510.html

Interoperability guidelines

Macro features within a digital repository is locally implemented via the technology used to build the digital repository or it makes use of external services. When the macro feature is locally implemented the references are provided within the adopted standards section. When a digital repository depends for the implementation of the macro feature on external services the interoperability guidelines are defined by the building blocks on which the external service is based.

Examples of solutions implementing this specification

In the table below, a non-exhaustive list of technologies which are used to build digital repositories is provided.

Technology	References
Fedora	http://fedora-commons.org/
Dspace	http://www.dspace.org/
Dataverse	https://dataverse.org/
Invenio	https://invenio-software.org/
B2SHARE	https://github.com/EUDAT-B2SHARE
EPrints	http://files.eprints.org/

Additional references

COAR Next Generation Repository

The COAR Next Generation Repositories Working Group published on the 28th of November 2017 a report on the results of this working group, including recommendations for the adoption of new technologies, standards, and protocols that will help repositories become more integrated into the web environment and enable them to play a larger role in the scholarly communication ecosystem. The report provides a definition of the next generation repository and described 11 new behaviours, as well as technologies, standards and protocols that will facilitate the development of new services on top of the collective network, including social networking, peer review, notifications, and usage assessment.

GEDE Repository Topic Group

The RDA Group of European Data Experts (GEDE) has a defined a working group to support interaction and discussion on the topic of digital repositories. The end result of the working group should be a report with agreed view on digital repositories among the GEDE experts. A draft version of this report can be found on the Repository Topic Group wiki space.

9.2 Metadata Management and Data Discovery

9.2.1 Metadata Cataloguing and Management

Metadata Cataloguing and Indexing comprises the entire metadata ingestion workflow, i.e.

- Metadata harvesting from community repositories

- Metadata mapping on common schema including curation and validation and

- Uploading and indexing of metadata records in the metadata catalogue, to enable *Data Discovery and Access*, see related macro feature.

High-level Service Architecture

The technical implementation of metadata cataloguing usually comprises five modules as shown in the figure below:

In the (Meta)data Provider Module, metadata must be available and harvestable in a known metadata schema and format. It also should be harvestable and accessible by a standardised transfer protocol (e.g. OAI-PMH).

For sustainable metadata ingestion synchronous and incremental harvesting should be set up on the service provider site.

On the service provider site, normalisation, homogenisation and mapping of the specific community standards onto a generic, common and unified metadata schema should be performed. The metadata mapping should be adopted to the needs of data provider and should include metadata validation and curation.

Finally, the mapped records are uploaded into the central metadata catalogue and indexed to allow faceted search in the discovery portal.

This enables now end users to search and filter datasets via the GUI or by using a command line tool and then access the found data resources.

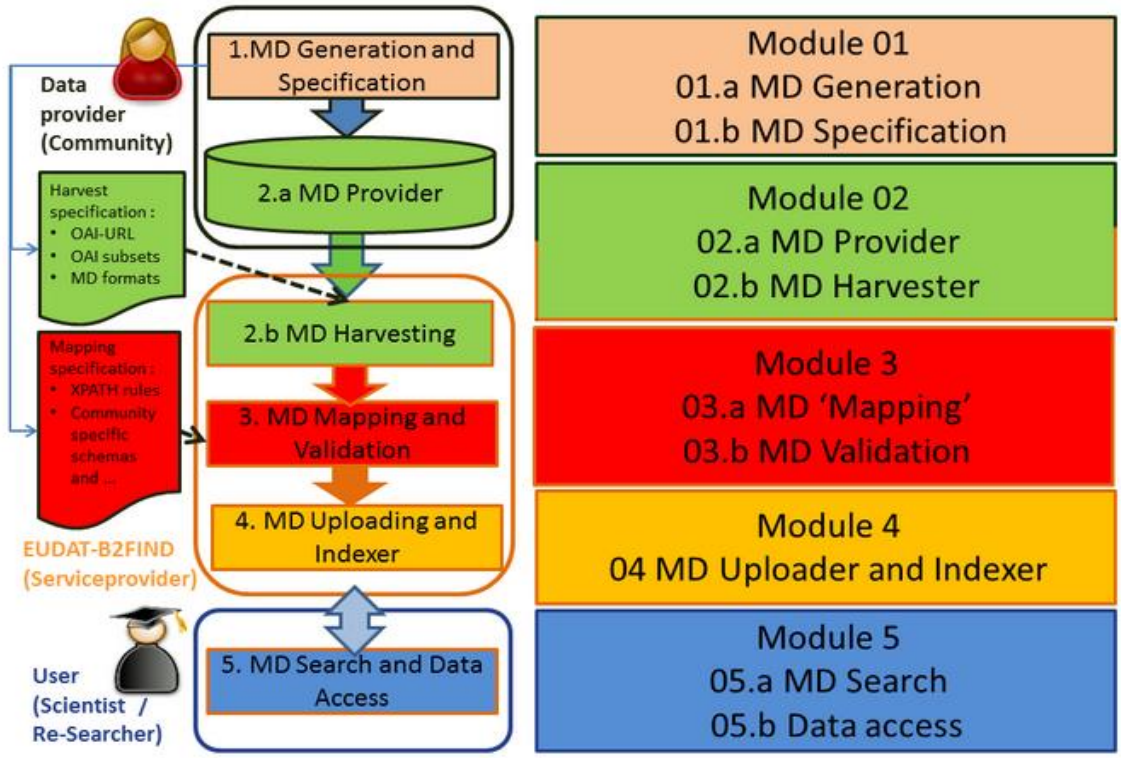


Figure 24: High level architecture diagram of Data Cataloguing and Management

Adopted Standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
Community specific metadata schemas and standards	Central, cross-domain Metadata aggregators collect community specific formatted metadata. For instance B2FIND supports harvesting of multiple metadata formats (as XML,MarcXML, JSON) and schemas (e.g. DataCite, Dublin Core, ISO 19115, CMDI, DDI and others).	A list of some domain specific metadata standards can be found at http://b2find.eudat.eu/guidelines/providing.html#mdformats
DataCite Metadata Schema 4.1.	Common and widely used Metadata Schema, on which e.g. OpenAire and EUDAT- B2FIND is based.	https://schema.datacite.org/meta/kernel-4.1/ http://b2find.eudat.eu/guidelines/mapping.html#b2fmdschema

Controlled Vocabulries	E.g. ISO 639-1 codes are a standardized nomenclature used to classify languages, or EUDAT-B2FIND develops a standardised taxonomy for 'Resaerch Disciplines' , which specifies the research disciplines	https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes https://cryptpad.fr/pad/#/1/edit/KDecbjauKCtZclOmZAbbWg/L4aEiGrzJlSbRSXrFutOb0Cd/ http://clara.science/

Protocol/API	**Short description**	**References**
OAI-PMH	The Open Archives Initiative Protocol for Metadata Harvesting provides an application-independent interoperability application to collect metadata from repositories.	http://www.openarchives.org/OAI/openarchivesprotocol.html
ResourceSync	This ResourceSync specification describes a synchronization framework for the web consisting of various capabilities that allow third-party systems to remain synchronized with a server's evolving resources.	http://www.openarchives.org/rs/1.1/resourcesync
REST API's	A full REST API is used to collect metadata formatted as JSON, e.g. the referenced REST API is used to 'harvest' from Herbadrop's repository	https://helpdesk.eudat.eu/Ticket/Attachment/122586/63597/RESTAPI_HowTo_SearchUserGuide_V3.pdf
CSW / OGC	Catalogue Service for the Web (CSW) is used to collect metadata from OpenGeoSpatial atalogues (OGC)	http://www.opengeospatial.org/standards/cat

Interoperability guidelines

In general the preconditions to publish metadata should be clearly described and stated by the discovery service provider in 'Guidelines for data providers', as in e.g. guidelines of OpenAire (https://guidelines.openaire.eu/en/latest/data/index.html) or of EUDAT-B2FIND (see http://b2find.eudat.eu/guidelines). This allows not only research communities, but also generic data storage repositories and metadata aggregators to make their data searchable in a simple way by following the guidelines.

Examples of solutions implementing this specification

Examples of cross-domain discovery services using this approach are:

- GoogleDataset Search (https://toolbox.google.com/datasetsearch), which crawls mainly schema.org , but supports no specific (meta)data curation and validation and does not consider on open data access (so 'dark data' is not necessarily excluded and it does not conform to FAIR data principles)
- EUDAT-B2FIND (http://b2find.eudat.eu/), the central indexer of EOSC-hub, provides an interdisciplinary discovery portal for research data with faceted search and comprises extensive meta(data) mapping, validation and curation in a FAIR manner.

Procedure to integrate a service with the EOSC Hub Metadata Cataloguing and Indexing

To provide metadata to the Metadata Cataloguing and Indexing service, the following preconditions must be fulfilled:

provider server must be set up (e.g. OAI-PMH provider)

Metadata must be provided in a standardised format and schema and made available and accessible for harvest requests and some mandatory fields (e.g a title and data identifier) must be provided.

In the next stage, refinement and enrichment of the metadata is done iteratively.

9.2.2 Data Discovery and Access

Data Discovery and Access comprises the ability for end-users to search for data resources and access the referenced data. This functionality requires and is based on the existence of an indexed metadata catalogue (see macro feature Metadata Cataloguing and Indexing).

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
DataCite Metadata Schema 4.1.	Common and widely used Metadata Schema, on which as well the B2FIND metadata schema and faceted search is based on	https://schema.datacite.org/meta/kernel-4.1/ http://b2find.eudat.eu/guidelines/mapping.html#b2fmdschema
ISO 639-1 codes	ISO 639 is a standardized nomenclature used to classify the search facet 'Language'	https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
B2FIND classification for Disciplines (not yet	Taxonomy for the central B2FIND facet *Discipline*, which specifies the research discipline the data belongs to. This allows filtering and selecting of datasets according to a multi-level discipline hierarchy	https://cryptpad.fr/pad/#/1/edit/KDecbjauKCtZclOmZAbbWg/L4aEiGrzJlSbRSXrFutOb0Cd/

standardized)		

Protocol/API	Short description	References
ElasticSearch	Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.	https://www.elastic.co/products/elasticsearch
CKAN-API	CKAN's Action API is a powerful, RPC-style API that exposes all of CKAN's core features to API clients.	https://docs.ckan.org/en/ckan-2.7.3/api/
SOLR	SOLR is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. SOLR powers the search and navigation features of many of the world's largest internet sites.	https://lucene.apache.org/solr/

High-level Service Architecture

The technical implementation of a data discovery and access service enabling searching for and identifying digital data should comprise the following components:

A discovery portal with an intuitive Graphical User Interface with faceted search and filtering options.

Command Line Interface allowing embedding discovery in a data processing workflow and machine readability.

A RESTful Search API with functionalities to identify referenced data collections by persistent identifiers

A search indexer and search index of a comprehensive metadata catalogue (see macro feature 'MD cataloguing')

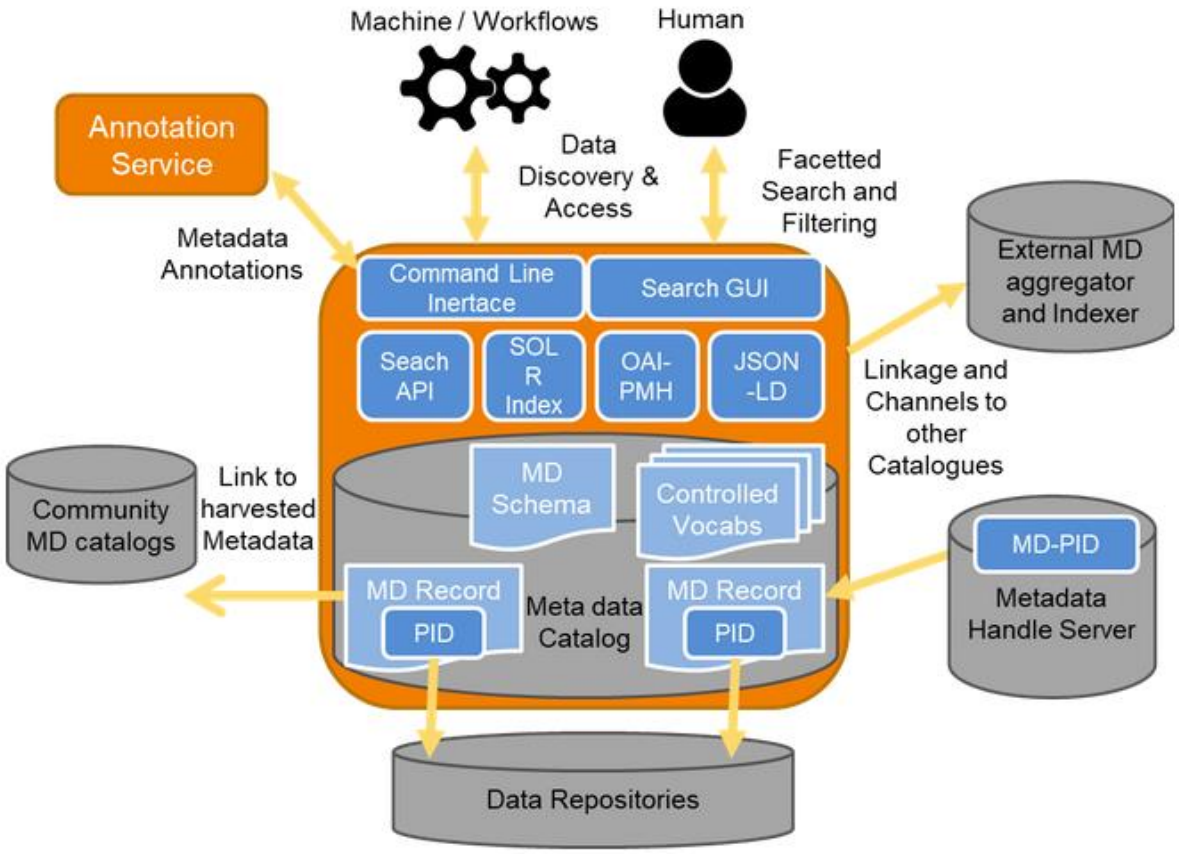


Figure 25: High level architecture diagram of Data Discovery and Access

Interoperability guidelines

General, how researchers can search for data via the GUI or the CLI is explained in a detailed search guide of the discovery services, e.g. search guides of DataCite (https://support.datacite.org/docs/datacite-search-user-documentation) or of EUDAT-B2FIND (see http://b2find.eudat.eu/help/searchguide.html). Often the CLI of discovery services are used to perform the first step of complex processing workflows, usually starting with identifying datasets, which serve as input for following data transfer, processing and storing tasks, executed by other services. Interoperability guidelines should show how the discovery workflow step can be integrated in such a processing chains. E.g. the CLI for B2FINd is implemented as python script (retrievable at https://github.com/EUDAT-B2FIND/md-ingestion/blob/master/searchB2FIND.py).

Examples of solutions implementing this specification

While there are countless domain-specific search portals and also many interdisciplinary discovery services, we will mention just two examples of cross domain services here:

- Google Dataset Search (https://toolbox.google.com/datasetsearch) allows users to find records stored on the Web using a simple keyword search. The tool can find information about records hosted in thousands of repositories across the Web. This makes these records generally accessible and usable.

- EUDAT-B2FIND (http://b2find.eudat.eu/) is a cross-domain discovery service based on metadata steadily harvested from research data collections from EUDAT data centres and other repositories covering all possible scientific fields. The service offers faceted browsing and it also allows, in particular, to filter via the facet 'Discipline' discovering data that is stored through the B2SAFE and B2SHARE services. The B2FIND service includes rich and validated metadata that is harvested from many different community and domain specific repositories. Within EOSC-hub EUDTA-B2FIND is intended to get the central search index for research date within and beyond EOSC-hub. Into this context fall the activities 'Integration of B2FIND with B2SAFE and EGI DataHub' already mentioned in 'Metadata Cataloguing and Indexing'

Procedure to integrate a service with the EOSC Hub Data Discovery and Access

The usual method to integrate discovery and access of data within other services is to add this function in a processing chain, which use other services. For example, using B2FIND, to search and identify datasets, which serve as input for services further down in the chain. The first workflow step 'Discovery of input data' can be implemented as a call of the python script searchB2IND.py with specified search criteria. A list of PIDs is then returned, which can be used to identify and retrieve data collections needed for further processing steps.

On the other hand, integration of other services in this context can mean, that the data of the associated provider are indexed and made searchable by the Discovery Service.

9.3 Cloud Compute, Containers and Orchestration

9.3.1 Cloud IaaS VM Management

Services of Cloud IaaS VM Management provide on-demand API-based access to computing resources as Virtual Machines that can run user-defined arbitrary software (including operating systems and applications). Services in this category also allow management of block storage that can be associated to the VMs and network management to provide connectivity between VMs and external networks.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
Open Virtualization Format (OVF)	Packaging format for software solutions based on virtual systems (VM image format)	OVF 2.1.1

There are several competing APIs for this block, most of them proprietary / closed and not interoperable. The main standards in the area (OGF OCCI and DMTF CIMI) have little support from vendors and/or providers and have little use. The table below lists some of the APIs implemented by IaaS providers, but it's not meant to be an exhaustive list.

Protocol/API	Short description	References
OpenStack	OpenStack is an Open Source cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed and provisioned through APIs with common authentication mechanisms.	OpenStack API
Amazon EC2/EBS/VPS & AWS VPN	Amazon Elastic Compute Cloud (EC2), Elastic Block Storage (EBS), Virtual Private Cloud (VPS) and AWS Virtual Private Network (AWS VPN) provide management of Virtual Machines and associated block storage and network features	AWS EC2 API
Azure Virtual Machines/Disks/VNet	IaaS VM management services from Microsoft Azure	Azure Virtual Machines API
Google Cloud Compute Engine	IaaS VM management service from Google Cloud Platform	Google Cloud Compute Engine API

High-level Service Architecture

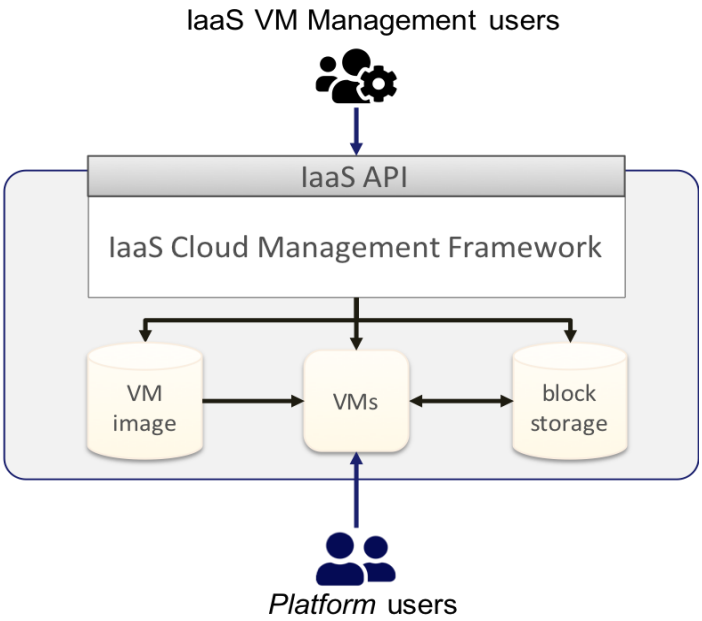


Figure 26: High level architecture diagram of Cloud IaaS VM Management

IaaS VM Management services allow users to manage VMs that are instantiated from VM images and can be associated with permanent block storage. The VMs can execute any kind of workload, including new services or platforms that are accessed by *platform* users, which may be different from the IaaS VM Management users that manage the IaaS resources.

Interoperability guidelines

Interoperable service in this category must:

- **Provide API access** for on-demand management of VMs and associated resources. Open and/or Standard APIs are preferred. Services that provide the capability to manage VMs through graphical dashboards but limit API access to users cannot be considered interoperable. See table above for a non-comprehensive list of APIs that may be supported by the service.

AAI interoperability

- Services should provide access to users authenticated with one of the EOSC-hub AAI federated identity protocols (OpenID Connect and/or SAML)

Orchestration interoperability

- Services should expose APIs that are supported by the IaaS Orchestrator services of EOSC-hub.

Federation interoperability:

- Services in this category that need to be federated into a cloud federation should provide API-based access to:
- Management of VM images, i.e. allow to create (upload) and delete VM images from which VMs can be instantiated.
- Access usage information of individual VMs and block storage so accounting records can be generated for integration into the EOSC-hub central services.

Examples of solutions implementing this specification Cloud IaaS VM Management

EOSC-hub services:

- EGI Cloud Compute - https://www.egi.eu/services/cloud-compute/

OpenSource implementations:

- OpenStack - https://www.openstack.org/
- OpenNebula - https://opennebula.org/

9.3.2 Cloud IaaS Container Management

Services of Cloud IaaS Container Management provide on-demand API-based management of container-based applications. These services support the (Automated) Orchestration of container-based applications which manage the deployment of a complete lifecycle of the containers that compose an application into a set of computing resources.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
OCI	OCI contains two specifications: the Runtime Specification (runtime-spec) and the Image Specification (image-spec). The Runtime Specification outlines how to run a "filesystem bundle" that is unpacked on disk. At a high-level an OCI implementation would download an "OCI Image" then unpack that image into an "OCI Runtime filesystem" bundle. At this point the "OCI Runtime Bundle" would be run by an "OCI Runtime".	OCI
Singularity Image Format (SIF)	SIF is the image format used by Singularity	SIF reference implementation

Protocol/API	Short description	References
Kubernetes	Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications	kubernetes
Docker Swarm	Cluster management and orchestration features embedded in the Docker Engine	Docker Swarm mode
Mesos	Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively.	Apache Mesos
AWS ECS	Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS.	AWS ECS
AWS Fargate	AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters.	AWS Fargate

High-level Service Architecture

IaaS Container Orchestration services allow users to manage applications that are composed by containers. The container orchestrator manages a set of bare-metal or IaaS Cloud resources where the containers are scheduled. The system also manages the containers lifecycle, associated storage for containers, provides networking among the application containers and exposes those as services to external applications, and scales up and down the deployments as needed. Other features may be also provided.

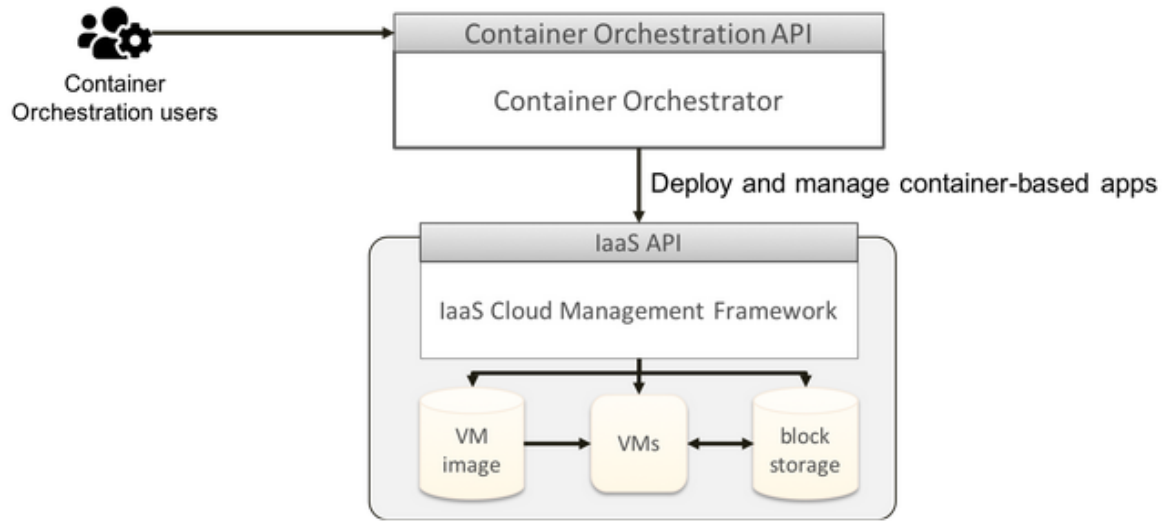


Figure 27: High level architecture diagram of Cloud IaaS Container Management

Interoperability guidelines

Interoperable service in this category should:

- Support OCI image and runtime specs for container execution.
- Provide access to users authenticated with one of the EOSC-hub AAI federated identity protocols (OpenID Connect and/or SAML).

If the service can manage underlying IaaS resources automatically, it should support main IaaS VM Management systems in the EOSC-hub (OpenStack mostly).

There are several non-compatible container orchestrators available, there are no guidelines for the APIs of those currently.

Examples of solutions implementing this specification

EOSC-hub services:

- EGI Cloud Container Compute

Other services:

- Kubernetes based: AWS EKS, GCP GKE, Azure AKS
- Other: AWS ECS, AWS Fargate

9.3.3 Cloud IaaS Orchestration

Services of Cloud IaaS Orchestration automate the deployment of resources on IaaS clouds. These tools normally use some sort of domain specific language or script that defines your application deployment process, that is translated to a set of tasks that interact with the cloud services to start Virtual Machines, Storage, Networks and other kinds of resources and services where your application will be installed and run.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
OASIS TOSCA	Topology and Orchestration Specification for Cloud Applications (TOSCA), is an OASIS standard language to describe a topology of cloud-based web services, their components, relationships, and the processes that manage them.	OASIS TOSCA

Protocol/API	Short description	References
IM	Infrastructure Manager is a tool that eases the access and the usability of IaaS clouds by automating the VMI selection, deployment, configuration, software installation, monitoring and update of Virtual Appliances.	IM REST API
Terraform	Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.	Terraform documentation
Occopus	Occopus is a framework that provides automatic features for configuring and orchestrating distributed applications (so called virtual infrastructures) on single or multi cloud systems.	Occopus
SlipStream	SlipStream is a multi-cloud application management platform.	SlipStream API

High-level Service Architecture

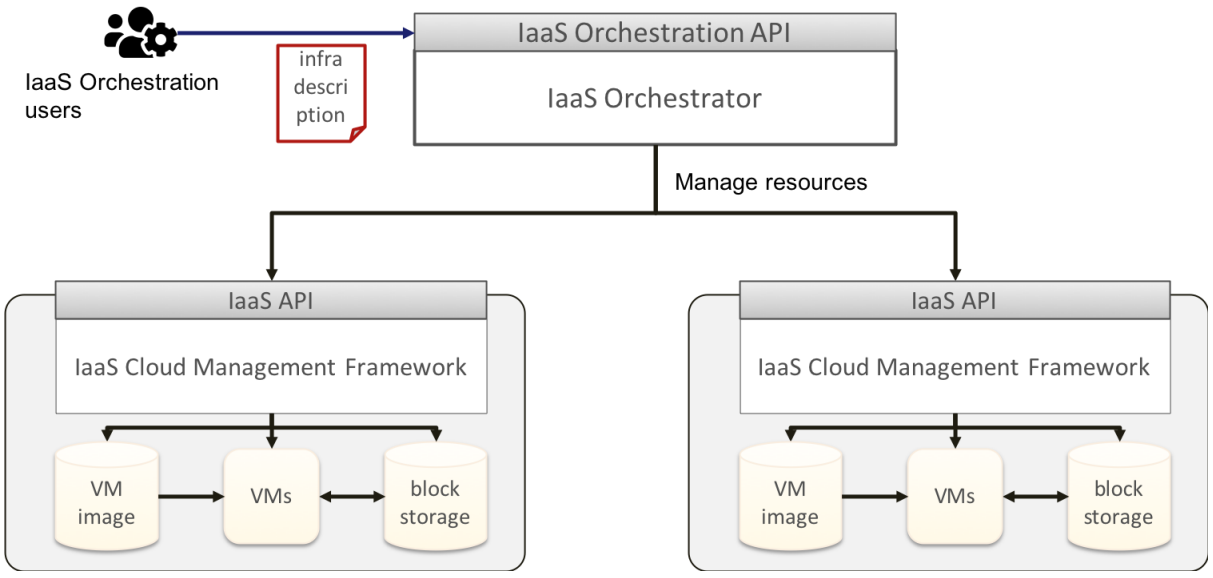


Figure 28: High level architecture diagram of Cloud IaaS Orchestration

IaaS Orchestration services build from a code-like description of an application the underlying resources by leveraging IaaS APIs of the target IaaS cloud providers.

Interoperability guidelines

Infrastructure description:

- Interoperable services should support a standard format for the description of the resources to be managed by the orchestrator. OASIS TOSCA is the most widely adopted standard in this area.

AAI interoperability

- Services should provide access to users authenticated with one of the EOSC-hub AAI federated identity protocols (OpenID Connect and/or SAML)

IaaS interoperability

- Services should support APIs listed in the IaaS VM Management macro-feature. At least the APIs supported by current EOSC-hub services (OpenStack, OpenNebula and OCCI) should be supported, ideally proprietary APIs should be also supported to avoid vendor lock-in from the underlying IaaS providers.

Orchestration API

- There is no clear standard or de-facto standard for the orchestration API. No interoperability guidelines are provided.

Examples of solutions implementing this specification

EOSC-hub services:

- Infrastructure Manager

Other:

- Terraform
- Occopus
- SlipStream

9.4 PaaS solutions

9.4.1 PaaS Orchestration

The PaaS (Platform as a Service) solution adopted in this project allows the users to deploy virtualised computing infrastructures with complex topologies (such as clusters of virtual machines or applications packaged as Docker containers) using standardized interfaces based on REST APIs

and adopting the TOSCA (Topology and Orchestration Specification for Cloud Applications) templating language for the description of Cloud-based applications.

The PaaS layer features advanced federation and scheduling capabilities ensuring the transparent access to the different IaaS back-ends including on-premises Cloud Management Frameworks such as OpenStack and OpenNebula, public Cloud providers such as Amazon Web Services and Microsoft Azure and, finally, Container Orchestration Platforms such as Apache Mesos and Kubernetes.

The selection of the best cloud provider to fulfill the user request is performed considering criteria like the user's SLAs, the services availability and the data location.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
TOSCA	OASIS open standard that defines the interoperable description of services and applications hosted on the cloud and elsewhere; including their components, relationships, dependencies, requirements, and capabilities, thereby enabling **portability** and **automated management** across cloud providers regardless of underlying platform or infrastructure.	http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.2/csprd01/TOSCA-Simple-Profile-YAML-v1.2-csprd01.html#_Toc503782167
Oauth2.0 Authorization Framework	The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.	https://tools.ietf.org/html/rfc6750
REST	REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are	https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest

	stateless and separate the concerns of client and server.	

High-level Service Architecture

The high-level **reference architecture** is depicted in the diagram below.

The architecture can be broken down into the following main categories of components:

- Core services:
  - *API server*, providing REST endpoints to submit and handle the deployment requests;
  - *Workflow Engine*, that manages the deployment workflow;
  - *Message Bus*, providing a way of integrating services loosely and based on notifications (events).
- Plugins
  - *Cloud connectors*, implementing the interfaces with the relevant Cloud Management Frameworks.
  - *Container orchestration connectors*, implementing the interfaces that abstract the interaction with the relevant container orchestration platforms, e.g. Mesos, Kubernetes.
  - *HPC integration connectors*, implementing the interfaces to interact with the HPC services; the envisage interaction is based on REST APIs provided by gateway hosted by the HPC site, e.g. using QCG APIs or SLURM APIs.
  - *Storage services connectors*, implementing the interfaces to interact with the relevant storage management services; the interaction is based on REST APIs provided by the storage services themselves.

Moreover, the following dependencies towards integration with the Federation Services are required:

- EOSC-hub AAI, to ensure the federated access to the services and resources;
- AppDB-IS or AMS (optional): information published by the sites can be used by the PaaS tools exploiting the already available collectors;
- EOSC-Hub Monitoring (optional): information about the health status of the services can be usefully exploited by the PaaS orchestrators in order to select the best sites for scheduling the user requests;
- Marketplace (optional): information collected in the Marketplace can be consumed by the PaaS tools.

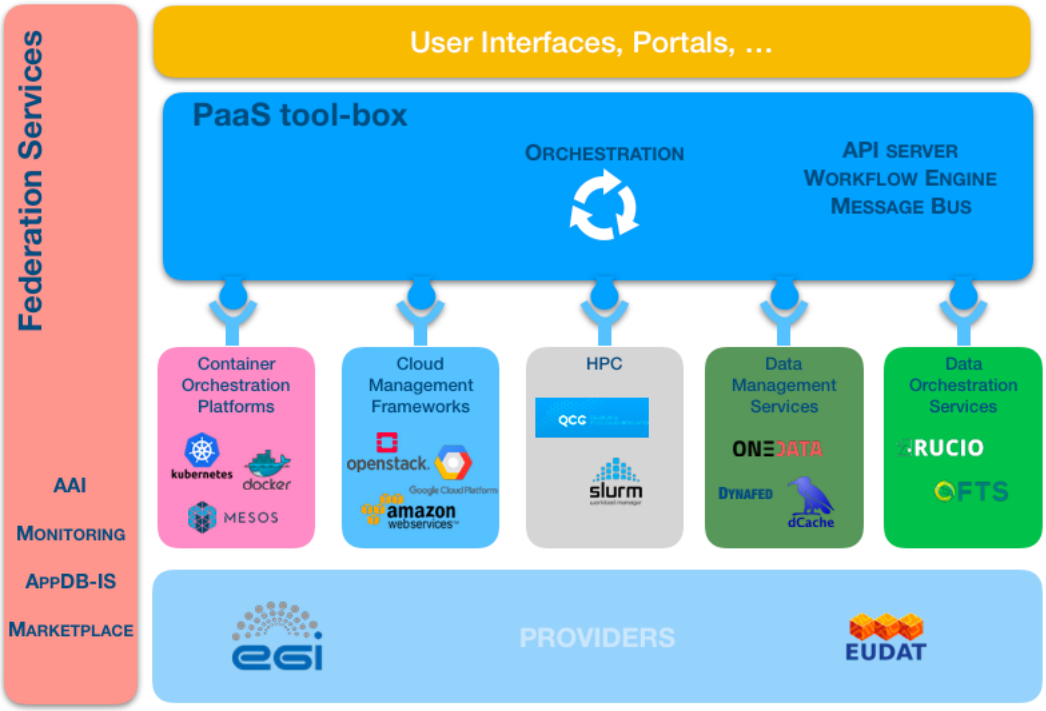


Figure 29: High level architecture diagram of the PaaS orchestration

Interoperability guidelines

The adoption of the TOSCA standard can help to reach a good level of interoperability among different services in this area. However, this is a necessary but not a sufficient condition since the full interoperability would require the adoption of the same TOSCA custom types (in addition to the normative ones) and of the same REST API specifications.

Currently there is not an official standard for the PaaS orchestration APIs, but we propose as reference the APIs implemented by the INDIGO PaaS Orchestrator: https://indigo-dc.github.io/orchestrator/restdocs/

Examples of solutions implementing this specification

- INDIGO PaaS Orchestrator: https://github.com/indigo-dc/orchestrator
- Infrastructure Manager: https://www.grycap.upv.es/im/index.php

9.5 Workflow Management and User Interfaces and Data Analytics

9.5.1 Marketplace

Marketplace is a dedicated platform where services are presented to the users and made available to get access to. Is a place where the Service Organisations can define and present to the users dedicated service offers, users can issue an order for those offers and handle different phases of the ordering process. Along with SPMT it is meant to support Service Management, and along with Service Order Management Back Office it provides Service Order Management in EOSC-hub.

Adopted standards

List with references of the main standards and protocols/APIs adopted by this core service

Standard	Short description	References
eInfraCentral based EOSC-hub STD	Service model. Unified approach to present services in the Marketplace, based on the MP-relevant subset of EOSC-hub STD.	https://github.com/eInfraCentral/docs https://wiki.eosc-hub.eu/display/EOSC/EOSC+hub+service+template
AAI user and group information attributes	User and group attributes served by the EOSC AAI (gathering EGI CheckIn, B2ACCESS, IAM) are recognised and processed in the MP.	https://documents.egi.eu/public/RetrieveFile?docid=3344&version=2&filename=EOSC-hub%20D5.1%20final.pdf#%5B%7B%22num%22%3A46%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C69%2C630%2C0%5D

Protocol/API	Short description	References
ARGO Messaging System - HTTP API	(Implements the Google PubSub protocol). MP uses this message-oriented service to retrieve list of EOSC-hub services along with their metadata (part of STD) relevant in the MP scope.	https://argoeu.github.io/guides/messaging/
eInfraCentral API - REST API	Used to retrieve information about services registered in eInfraCentral Catalogue	https://github.com/eInfraCentral/docs/blob/master/eInfraCentral_APIs_v1.0.pdf
JIRA webhooks	User-defined callback over HTTP. Used to retrieve information from JIRA in the scope of MP Projects (represented by JIRA Epic) and MP Orders (JIRA tasks)	https://developer.atlassian.com/server/jira/platform/webhooks/

JIRA REST API	Used to create and synchronise representation of MP Projects and Orders in JIRA. Data representing Projects and Orders on the MP end is available in JSON format.	https://developer.atlassian.com/server/jira/platform/rest-apis/

High-level Service Architecture

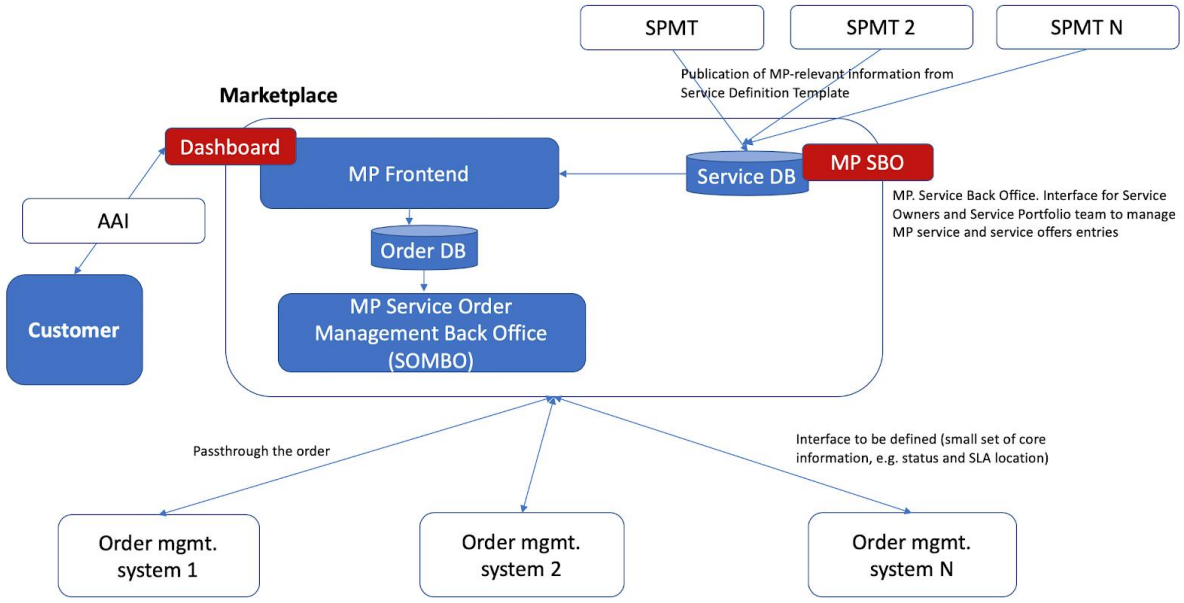


Figure 30: High level architecture diagram of Marketplace

Interoperability guidelines

For SPMT and Order Management solutions, a dedicated development is needed to achieve the integration. However, in the future it is planned to have MP API to integrate with SPMT solutions on one end and JIRA-like solutions on the other (two APIs) with a unified approach to exploit the core service features.

There is a standard approach to integrate JIRA based Order Management solutions. It requires a dedicated JIRA instance, allowing webhook integration and information about the configuration of JIRAs objects (IDs of JIRA ticket fields, ticket statuses, worklows info etc.) in order to configure the mapping between the MP and JIRA.

Examples of solutions implementing this specification

SPMT solutions:

- EOSC-hub SPMT: AGORA SPMT (https://grnet.github.io/agora-sp)
- eInfracentral Catalogue (https://github.com/eInfraCentral/docs)

Order Management solutions:

- JIRA based EOSC-hub Order Management System (https://docs.google.com/document/d/1o1pu7_3tVHuYmY9AcvZL9PGS-xIZYa_bcyYCyQQQ5HU/edit#heading=h.amnmdg2nez9w)

AAI solutions:

- EGI CheckIn based EOSC AAI (https://documents.egi.eu/public/RetrieveFile?docid=3344&version=2&filename=EOSC-hub%20D5.1%20final.pdf#%5B%7B%22num%22%3A46%2C%22gen%22%3A0%7D%2C%7B%22name%22%3A%22XYZ%22%7D%2C69%2C630%2C0%5D)

Procedure to integrate a service with the EOSC Hub Marketplace

For SPMT and Order Management solutions a dedicated development was needed to achieve the integration. However, in the future it is planned to have MP API to integrate with SPMT solutions on one end and JIRA-like solutions on the other (two APIs) with a unified approach to exploit the core service features.

For the AAI integration, MP used a defined procedure of EGI CheckIn to integrate SPs/IdPs (https://wiki.eosc-hub.eu/pages/viewpage.action?pageId=30738897)

10 Conclusions and next steps

This deliverable presented the EOSC-hub proposal for the EOSC Technical Architecture that consists of the definition of a reference architecture, where all the EOSC main functions, interfaces, APIs and standards are identified. A common approach to identify key technical functions/building blocks for service category has been defined and started to be applied. As a result, several building blocks have already been identified and technical specifications are available for some of them.

As a next step, we have started a process to share our approach and collecting feedback. Initially a webinar was held where this work has been presented, then a formal feedback collection will start in the next few weeks and we are planning to organise a workshop by the end of this year involving the largest expected EOSC user groups.

Feedback is also needed on the technical specifications we are defining with the expertise available within the project. For every specification, we are intending to open a forum with technical experts from other initiatives with the double aim to improve the specification and find consensus around it.

Finally, we will liaise with the EOSC Architecture WG to continue this activity taking into account suggestions and requirements from the EOSC governance and the largest possible set of service providers and user communities.