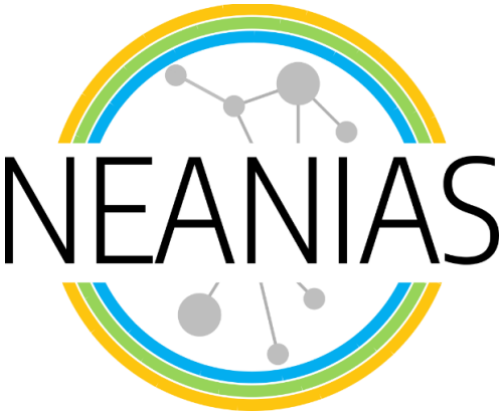


NEANIAS
Novel EOSC services for Emerging Atmosphere, Underwater and Space Challenges

Deliverable Report

Deliverable: D6.1 Core Services Architecture, Design Principles and Specifications

30/04/2020

NEANIAS is funded by European Union under Horizon 2020 research and innovation programme via grant agreement No. 863448

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Document Info

Project Information			
Acronym	NEANIAS		
Name	Novel EOSC Services for Emerging Atmosphere, Underwater & Space Challenges		
Start Date	1 Nov 2019	End Date	31 Oct 2022
Program	H2020-EU.1.4.1.3. - Development, deployment and operation of ICT-based e-infrastructures		
Call ID	H2020-INFRAEOSC-2018-2020	Topic	H2020-INFRAEOSC-2019-1
Grant No	863448	Instrument	RIA

Document Information			
Deliverable No	D6.1		
Deliverable Title	Core Services Architecture, Design Principles and Specifications		
Due Date	30-APR-2020	Delivery Date	12-MAY-2020
Lead Beneficiary	INAF		
Beneficiaries (part.)	ATHENA, ALTEC, MEEO, JUCOBSUNI, UNIMIB, UOP, CITE, CORONIS, SZTAKI, INAF, NKUA, GARR		
Editor(s)	Eva Sciacca (INAF), Georgios Kakaletris (CITE)		
Authors (s)	Eva Sciacca (INAF), Georgios Kakaletris (CITE), Georgios Papanikos (CITE), Nikos Chondros (NKUA), Michalis Konstantopoulos (NKUA), Claudio Pisa (GARR), Eugenio Topa (ALTEC), Rosario Messineo (ALTEC), Angelo Fabio Mulone (ALTEC), Mel Krokos (UoP), Giuseppe Vizzari (UNIMIB), Gabor Kertesz (SZTAKI), Jozsef Kovacs (SZTAKI), Attila Farkas (SZTAKI), Robert Lovas (SZTAKI), George Papastefanatos (ATHENA), Christina Peraki (ATHENA)		
Contributor (s)	Cristobal Bordiu, Ugo Becciani (INAF), Diamantis Tziotzios (CITE), Boutsis Spyridon (CITE), Konstantinos Kakaletris (CITE), Josep Quintana (CORONIS), Ricard Campos (CORONIS)		
Reviewer(s)	Paul Wintersteller (UBREMEN), Christian Dos Santos Ferreira (UBREMEN)		
Workpackage No	WP6 Core Services Foundation and Implementation		
Version	V1.0	Stage	Final
Version details	Revision: *14* . Last save: *2020-05-13 , 18:39* Pages: *104* . Characters: *160.706*		
Distribution	Public	Type	Report
Keywords	Services; Architecture; EOSC; Service Oriented Architecture;		

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Change Record

Version	Date	Change Description	Editor	Change Location (page/section)
1.0	12/5/2020	Document version submitted to EC	Eva Sciacca, Georgios Kakaletris	

Disclaimer

NEANIAS is a Research and Innovation Action funded by European Union under Horizon 2020 research and innovation program, via grant agreement No. 863448.

NEANIAS is project that comprehensively addresses the 'Prototyping New Innovative Services' challenge set out in the 'Roadmap for EOSC' foreseen actions. It drives the co-design, delivery, and integration into EOSC of innovative thematic services, derived from state-of-the-art research assets and practices in three major sectors: underwater research, atmospheric research and space research. In each sector it engages a diverse set of research and business groups, practices, and technologies and will not only address its community-specific needs but will also enable the transition of the respective community to the EOSC concept and Open Science principles. NEANIAS provides its communities with plentiful resource access, collaboration instruments, and interdisciplinary research mechanisms, which will amplify and broaden each community's research and knowledge generation activities. NEANIAS delivers a rich set of services, designed to be flexible and extensible, able to accommodate the needs of communities beyond their original definition and to adapt to neighboring cases, fostering reproducibility and re-usability. NEANIAS identifies promising, cutting-edge business cases across several user communities and lays out several concrete exploitation opportunities.

The European Union (EU) was established in accordance with the Treaty on the European Union (Maastricht). There are currently 28 member states of the European Union. It is based on the European Communities and the member states' cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice, and the Court of Auditors (http://europa.eu.int/).

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Table of Contents

Document Info **2**
Change Record **3**
Disclaimer **4**
Table of Contents **5**
Tables of Figures & Tables **8**
Abstract **9**
1. Introduction **10**
1.1. Context 10
1.2. Contents and Rationale 10
1.3. Structure of the document 11
2. Design Principles **12**
2.1. Service Oriented Architecture 12
2.2. Minimum service requirements 13
2.3. REST paradigm 14
2.4. Pluggability & Extensibility 15
2.5. Interoperability 16
2.6. Standards 17
2.7. F.A.I.R. principles 18
2.7.1. C1 core services 18
2.7.2. C2 core services 19
2.7.3. C3 core services 20
2.7.4. C4 core services 20
2.8. Operation 20
2.9. Security 20
2.9.1. NEANIAS AAI Concepts 21
2.9.2. Policy 24
2.10. User Interfaces / User Experience 24
2.11. Portability 25
2.12. Technology 26
2.12.1. Programming languages 26
2.12.2. Development frameworks 26
2.12.3. Data Management frameworks 26
2.12.4. AI/ML frameworks 27
2.12.5. Visualization Frameworks 27
2.12.6. Major Background Systems 27

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

3. System Architecture 29
3.1. Architecture overview 29
3.2. Logical architecture 30
3.3. Fundamental building blocks 32
3.4. NEANIAS reference service 34
3.5. Physical architecture 37
4. Fundamental resource abstractions in NEANIAS 40
4.1. Storage 40
4.2. Computation 40
4.3. Other resources 41
5. C1 Open-Science lifecycle support services & Components reference 42
5.1. The role of C1 services in NEANIAS 42
5.2. NEANIAS Service Catalogue 42
5.3. NEANIAS Research Product Catalogue 43
5.4. Data Validation Service 44
5.5. Common User Interface Components 45
5.6. NEANIAS Access Gate 46
5.7. OpenDMP / Argos 47
5.8. Data Publishing Service 48
5.9. Persistent Identifier Service / Zenodo 49
6. C2 EOSC hub, RIs and cloud integration enabling services reference 50
6.1. The role of C2 services in NEANIAS 50
6.2. NEANIAS AAI 50
6.3. Configuration Management Service 53
6.4. Service Instance Registry 54
6.5. Log Aggregation Service 56
6.6. Accounting Service 57
6.7. Notification Service 60
6.8. Data Depositing service 62
6.9. Data Sharing service 63
6.10. Data Transfer service 64
6.11. Data exploration service 66
6.12. Computational resources access service 67
7. C3 Artificial Intelligence services reference 73
7.1. The role of C3 services in NEANIAS 73
7.2. C3.1 AI Science Gateway: service for development of ML models using Jupyter Hub 73
7.3. C3.2 Serving trained ML models 75
7.4. C3.3 Distributed Multi-GPU training of large ML models using Horovod 77

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

7.5. C3.4 Distributed Machine Learning using SparkML 78

8. C4 Visualisation services reference 80

8.1. The role of C4 services in NEANIAS 80

8.2. C4.1 Framework for Visual Discovery (VD) 81

8.3. C4.2 Visualisation Gateway (VG) 85

8.4. C4.3 Toolkit for Cross Realities (XR) 88

8.5. C4.4 Spatial Data Stores (DS) 90

9. Workplan – Timeline 93

10. Conclusions 101

List of acronyms 102

Appendix 1 – Service Description Template 103

Tables of Figures & Tables

Document Figures

Figure 1: Logical layering of NEANIAS service clusters **30**
Figure 2: Logical architecture diagram **32**
Figure 3: Reference service instantiation lifecycle **35**
Figure 4: Reference servicing lifecycle **37**
Figure 5: Hypothetical deployment of NEANIAS services **38**

Document Tables

Non è stata trovata alcuna voce dell'indice delle figure.

Abstract

The objective of deliverable D6.1 is to report on architecture, specifications and software development plan of the Core Services that will be developed in WP6. This document reports on the general design principles, the overall system architecture and fundamental resources. Then it presents the foreseen core services related to manage Open Science lifecycle (C1), EOSC integration (C2), Artificial Intelligence processing (C3) and, finally, Visualization (C4).

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

1. Introduction

1.1. Context

The NEANIAS WP6 “Core Services Foundation and Implementation” establishes generic, cross-community services that amplify the potential of thematic services delivered by WP2-5, enabling Open Science, and facilitating the migration to the EOSC concepts, by streamlining access to cloud resources.

In particular Task 6.1 “Service gap analysis and specifications” builds on the requirements delivered by the community sector WPs (WP2, WP3 and WP4) and those delivered by innovation cases in WP5. It also utilizes the information gathered by WP8 on EOSC hub landscape and trends and consortium knowledge and acquaintance with Research Infrastructures (RIs). It locates and highlight the gap from Open Science principles as augmented by the vision of NEANIAS, and deliver specifications that bridge the gap. The task is also responsible for drafting an implementation roadmap with consolidated contribution from all other tasks of the WP, aligned to the milestones and to the research sector development plans, able to achieve delivery of software artefacts in due time for service delivery by WP7.

Task 6.2 “Architecture and interoperability approach design and validation” builds on outcomes of T6.1 and input coming from other work packages and yields the architecture of NEANIAS framework and the set of its service set, covering both, core and thematic ones, towards a holistic architectural alignment of its services. It sets the principles for interoperability of services, both internally as well as externally. With a strong preference in well-established standards, it presents options that services shall adopt for maximising their interoperability and reuse opportunities. Core service implementation tasks will be based on outputs of T6.1 and T6.2, aligning TRL6 existing services or EOSC Hub services to the NEANIAS framework.

Two core services (C1 and C2) are dictated by the needs and opportunities raised in EOSC. C1 includes research lifecycle empowering services, providing essential tools for registering, locating, inspecting, sharing data and services and C2 allows for EOSC/RIs Integration services for the exploitation of EOSC and RIs offerings in a unified manner for NEANIAS services, including access to storage and computation and mechanisms for authentication and authorization. Abstracting on thematic service concepts and analyzing the opportunities of innovation on EOSC, the other two services (C3 and C4) will be the arrowhead of NEANIAS generic service offerings: C3 includes AI services, that enclose also capacities for Machine Learning and C4 offers Visualisation services that deliver state-of-the-art visualization as a service.

1.2. Contents and Rationale

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

This deliverable D6.1 reports the core services design principles architecture, and specifications led by Tasks 6.1 and 6.2 including a detailed description and implementation plan for C1/C2/C3/C4 core services developed in Tasks 6.3, 6.4, 6.5 and 6.6.

1.3. Structure of the document

In Section 2 the services design principles are presented, those include the Service Oriented Architecture paradigm guidelines, REST paradigm, Standards and Interoperability guidelines. Section 3 describes the overall NEANIAS System Architecture including the logical architecture, a generic service processing lifecycle and the proposed physical deployment. Section 4 focuses on the NEANIAS fundamental resources abstractions, namely storage and computation. Chapters 5, 6, 7, 8 presents the Core Services C1, C2, C3 and C4 respectively, detailing, in tabular format, all the main information regarding e.g. technologies adopted, dependencies with other core services, licensing etc. A work plan of the services delivery together with a proposed timeline is presented in Section 9. Finally, Section 10 provides conclusions and plans for the future activities.

2. Design Principles

This section reports the design principles of the Core Services and overall NEANIAS architecture based on Service Oriented Architecture (including the list of minimum services required), REST (REpresentational State Transfer) paradigm, FAIR principles (to make data Findable, Accessible, Interoperable, and Reusable) and User Interface guidelines. This section also details on main aspects regarding pluggability, extensibility, interoperability, portability, security, operation and technology.

2.1. Service Oriented Architecture

Service Oriented Architectures (SOA) are driven by several principles that although varying in number have a core notion of service reuse, separation, discoverability, interoperability etc.

NEANIAS adopts the Service Oriented Architecture paradigm guidelines, with varying degree of enforcement and the addition of microservices. Furthermore, it adds its own principles for services matured and delivered in the context of NEANIAS.

Essential motivations behind the adopted approach are the following:

- NEANIAS builds on existing services of substantial maturity that act in coordination with other systems, implying a completely different paradigm of software architecture.
- Services in the context of NEANIAS and EOSC do not adhere to the notion of services as expressed in the SOA paradigm. As such their granularity, model of operation and interaction is quite different than in common SOA services.
- A subset of NEANIAS infrastructure requires a level of integration among specific services which in cases may compromise the loosely coupling and isolation principles.
- NEANIAS policy with respect to service dependencies promote standards vs custom service contracts and attempts to isolate services sustainability risks with a number of techniques that drive the technological choices behind specific lower level infrastructure services.

In the following we briefly present the SOA principles adopted in NEANIAS:

- **Contracts**: Services will present clear definitions of their interfaces and their expected interaction patterns that they will respect. In NEANIAS the following policies shall be respected:
    - The primary candidate interface form for services shall be compliant with REST, although reasonable exceptions may apply.
    - Services shall respect their interface contracts and apply to the best of abilities backwards compatibility among their versions. Compatibility shall be clearly declared in their contract
    - Where well known or defacto standards apply for interfaces of specific services, those will be adopted.
- **Discoverability**: NEANIAS services shall be registered in central repository so that they can be found and invoked by other services. The principle is enforced for top level services or services that are enabling the infrastructure; however, it is not imposed for microservices that might be supporting a service. Furthermore, static and dynamic

registration are both valid approaches due to the assumptions of specific services. All services will start from 0-point assumptions declared in their base configuration. In NEANIAS there will be the following classes of services with respect to discoverability:

  - **Fixed EndPoint Services:** Services that respond to a predefined endpoint. Registration may be performed statically and is optional.
  - **Dynamic EndPoint Services:** Services that register themselves into the infrastructure service instance catalogue upon instantiation and need to be discovered in order to be exploited.

- **Loose Coupling**: Services shall not be making any further assumption on their dependencies other than the ones declared in their interaction contract, so that the services may be substituted by other services. The principle is suggested yet not enforced in NEANIAS as there are several services that make deep assumptions on microservices they utilize, which themselves may be independent services. Loose coupling is a strong asset for service sustainability as it helps reduce risks. However, in NEANIAS it is also assumed that most services will conform to a few internal behavioral contracts. Those relate to security, accounting, logging etc
- **Encapsulation**: Services should not be exposing to their consumers details on how they perform their tasks. However, this is not a strongly imposed principle as services may be assuming specific features of services they depend on. It is suggested that services expose their internal features that clients may wish to depend on, in the registration service, so that they can be explored by clients.
- **Reusability**: Services in NEANIAS are suggested to be defined in a manner that maximizes reusability. This is essential for all major identified services of the ecosystem; however, it is not enforced for microservices underlying those which may be tailored to specific tasks.
- **Statelessness**: Services, to the best of their efforts shall assume no state maintenance among their invocations. However, this may not be semantically feasible as services may be depending on data that grow via subsequent invocations and although stateless in their interface, they are not stateless internally. Supporting services will also need to handle state in more than few examples. Thus, although a good practice statelessness is not enforced. In technical terms NEANIAS promotes the REST paradigm for service interactions which further supports this directive.
- **Autonomy**: Services have control on the logic they encapsulate and define their terms of operation. The principle is not generally adopted by NEANIAS but may be followed by specific services for their specific mission critical and, in exceptional occasions, sustainability reasons.
- **Composability**: Services should be allowing callers to compose them into larger logic workflows to carry out composite tasks. The principle is not only strongly suggested in NEANIAS, but it is also supported by several architecture design choices that allow services to be composed building on assumptions on fundamental concepts, such as security.

2.2. Minimum service requirements

All NEANIAS services **must conform** with the following requirements:

- Present their users with a clear and definite **Terms of Use** text that defines intended use and clarifies how the service may be used, presents its SLA and the data protection rules applied, and if applicable exemplifies and discourages potential malicious use.
- Adopt the **Authentication Authorization Infrastructure** (AAI) policies and protocols established in NEANIAS core AAI and conform to one of the AuthN/AuthZ models introduced in this report and follow its evolution in the course of project's agile design and development activities.
- Generate all **accounting** information required for the infrastructure, among others, to evaluate the use of resources, calculate KPIs and potentially, in the future, apply quotas to resource usage.
- Take all **security** measures to:
    - Comply with the General Data Protection Regulation (GDPR) and the policies settled by the project.
    - Respect and enforce data protection assumptions made and declared to end users.
    - Prevent infrastructure from malicious use, especially from its own internal logic and to its best effort by its users.
    - Utilize resources placed at their disposal in compliance with resource provider's policies.
- Respond to handling any major direct or indirect (i.e. dependencies) security defects identified by other partners and 3rd parties.

Furthermore, NEANIAS services are **encouraged** to:

- Provide standard logging information that ships to central log analysis facility for consolidated log analysis and troubleshooting.
- Seek and adopt official or de fact standards for
    - data they generate and/or consume
    - any interaction with other services, unless such specifications are outdated and do not comply with modern technologies and service needs.
- Publish their output data in the catalogues of the infrastructure:
    - Register metadata in catalogues, presenting enough evidence for discovery but also identification the origin and process of the data they refer to.
    - Register data in long term data repositories suggested by the project
- Adopt the most portable supported model suitable for their operation and needs.

2.3. REST paradigm

NEANIAS services as well as the existing EOSC services are employing well-established web technologies, i.e. HTTP REST for the implementation of their API methods. An API (Application Programming Interface) is a protocol intended to be used as an interface by software components to communicate with each other. **RE**presentational **S**tate **T**ransfer (**REST**) is an architectural style, or design pattern, for APIs. REST specifies a few architectural constraints that must be satisfied for an API to be referred to as RESTful. These constraints, such as client-

server stateless interactions, cacheable resources, layered system and uniform interface, when applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. When a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource. The resources are acted upon by using a set of simple, well-defined operations. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol. The REST architectural style is designed to use a stateless communication protocol, typically HTTP. HTTP defines a set of request methods to indicate the desired action to be performed for a given service resource. Each of these methods (referred to as HTTP verbs) implements a different action. The primary or most-commonly-used HTTP verbs are POST, GET, PUT, PATCH and DELETE. These correspond to create, read, update, and delete (or CRUD) operations.

Some common design best practices used when designing REST APIs are:

- Use of the JSON standard for transferring data. There are other ways to transfer data as for example XML, however they are not as widely supported by frameworks.
- Use of nouns instead of verbs for defining resources that make sense from the perspective of the API consumer. This is preferred because HTTP request methods already include verbs. For example, a resource could be called “users” and the action GET /users could retrieve a list of all users.
- Use of standard HTTP error codes as response codes when an error occurs.
- Use of SSL/TLS for security for the REST APIs to communicate over secure channels and protect information exchanged.
- Use of API versioning for API updates in order to make the transition to new versions smoothly and prevent invalid requests to updated endpoints.
- Use of data caching to improve performance.
- Use of data filtering, sorting and pagination to improve performance especially when there is too much data to be returned all at once.

2.4. Pluggability & Extensibility

Pluggability and extensibility are key design principles for building EOSC services for NEANIAS. For the long-term development strategy of a service, the capability for functional updates and extensions must be provided in order to implement the newly arising users' demands in a seamless way.

While extensibility enables developers to expand or add capabilities of a service without modifying the original source code, pluggability is one of the possible approaches to support

it via a pluggable framework and plugins. The two together are the capability to add new modules to a service without recompiling or even restarting a running service.

Modularity is an important aspect of the service architecture in order to serve pluggability and extensibility with low development effort. In order to provide this, the application of microservices architecture design is highly recommended. In microservice architecture, multiple loosely coupled services work together. Each service focuses on a single purpose and has a high cohesion of related behaviors and data.

The main motivations for microservices architecture to support pluggability and extensibility are as follows:

- Modularity: Each microservice represents a logically cohesive, lightweight and independent business functionality with well-defined boundaries. By design, microservices are highly granular, and independently built and deployed.
- Loose coupling: Microservices are designed to be loosely coupled with minimal dependency on other services and libraries.
- Extensibility: Microservices can be leveraged to create an extensible solution by quickly onboarding newer ones.
- Communication: Microservices can be effectively built by using communication standards such as REST.

Communication between microservices and with client applications has to happen fast with low overhead (e.g. no server-side session management and lean message structure) and network latency, REST APIs are a good fit. Beyond performance, applying RESTful API is also a key factor to support the aforementioned key features. For details on REST API, please see Section 2.3.

2.5. Interoperability

Interoperability is usually considered as the ability to interconnect data and e-infrastructures. The key to exchange data among components in an e-infrastructure is the utilization of standards in communication and data format.

Interoperability is one of the most important aspects in EOSC, where the aim is to build a huge European-wide e-infrastructure for scientists to exchange data, services and resources. In order to realize this goal, the EOSC Pilot[1] project investigated the most typical reasons for non-interoperability happening in e-infrastructures (see link below). The reasons are identified and named by 6 different gaps along with 6 risks behind the gaps. To lower the risks preventing infrastructures' interoperability the EOSC pilot project defined 14 recommendations to ensure infrastructure's interoperability. Finally, 6 recommendations were stated to adapt FAIR data principles for the EOSC.

[1] https://eoscpilot.eu/

The FAIR Data Principles are guidelines to make data findable, accessible, interoperable and reusable. However, concerning EOSC, it is important to complement the FAIR principles with further recommendations that improve the availability of research data to users and services through an open cloud infrastructure. These recommendations can be found at https://eoscpilot.eu/eoscpilot%E2%80%99s-contributions-eosc-interoperability. Section 2.7 will detail how these principles will be implemented in the core services.

2.6. Standards

All core services will be delivered according to standards, on the direction to support the targeted research sectors (WP2, WP3, WP4) and communities as well as the targeted additional business cases (WP5).

NEANIAS vision on EOSC is the full compliance with capable and prominent standards and widely adopted specifications, as the only road towards wide and undoubtable interoperability means. Thus, it plans to not only adopt but also, where needed, to propose augmentation of models and specifications, promote the use by engaged stakeholders and where ground is found to propose new approaches capable of being standardized in the future. The establishment of strategic synergies with relevant national, European and international initiatives, is essential element to achieve this impact; those synergies will target both the reuse of tools and services from EOSC hub, Research Infrastructures and other domains, based on commonly agreed protocols, and the establishment/expansion of working groups in areas of interest. Research Data Alliance (RDA) participation in areas such as Array Database Assessment, Data Management Plan (DMP) Common Standards, as well as groups dealing with data identification, description, provenance and quality will be actively followed.

As already mentioned in Deliverable D9.1, NEANIAS partners are significantly linked to several standardization consortium and organizations toward sustainability and interoperability, such as the Open Geospatial Consortium (OGC), the Open Navigation Surface (ONS), the International Virtual Observatory Alliance (IVOA), the Internet Engineering Task Force (IETF) and the OpenID Foundation (OIDF).

Some of the standards that will be adopted in NEANIAS services are as listed below, but the list is not meant to be exhaustive and is expected to be extended during the development:

- IETF OAuth2: An open standard for authentication / authorization.
- OpenID: An open standard for authentication.
- OGC WMS: An open standard for rendering of digital maps composed of raster and vector data.
- OGC WFS: A standard interface allowing requests for geographical features across the web using platform-independent calls.
- OGC WCS: An Interface Standard that defines Web-based retrieval of digital geospatial information representing space/time-varying phenomena.
- OGC WPS: An Interface Standard that provides rules for standardizing inputs and outputs, and relative requests and responses, for geospatial processing services.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- OGC WMTS: A standard protocol for serving pre-rendered or run-time computed georeferenced map tiles.
- OGC GeoTIFF: An open standard managing the Tagged Image File Format (TIFF) for the exchange of georeferenced or geocoded imagery.
- OGC GeoPackage: An open, non-proprietary, platform-independent and standards-based data format for geographic information system implemented as SQLite database container.
- OGC 3D Tiles: A standard for streaming and rendering massive 3D geospatial data using tiled hierarchical data structures.
- IVOA SAMP: An open standard messaging protocol that enables astronomy software tools to interoperate and communicate.
- IVOA TAP: An open standard interface providing a general access mechanism for tabular data, including but not limited to astronomical catalogs.
- IVOA VOTable: An XML based standard for the interchange of data represented as a set of tables, with particular emphasis on astronomical tables.
- FITS: An open standard format and transport system used in astronomy data.
- OSGeo TMS: A specification for tiled web maps.
- ONS BAG: A file format designed to store and exchange bathymetric data.
- Unidata NetCDF: A community standard for sharing array-oriented scientific data.
- OAI-PMH: A protocol for harvesting metadata descriptions of records in an archive so that services can be built using metadata from many archives.
- EOSC-EDMI[2]: a minimum information metadata guideline defined by EOSCpilot to help users and services to find and access datasets reusing existing data models and interfaces.

And many other standards related to the W3C recommendations and protocols such as HTML5/JavaScript/CSS3, JSON, XML, HTTP(S) and FTP.

2.7. F.A.I.R. principles

This section describes how the principles to make data Findable, Accessible, Interoperable and Re-usable (FAIR) will be supported by each of the NEANIAS core services belonging to C1, C2, C3, and C4. Those details complement information included in D1.5 presenting the overall NEANIAS Data Management Plan.

2.7.1. C1 core services

FAIR data principles will be followed by all C1 core services.

Findable data. Access to the service catalogue and service metadata will be provided through the NEANIAS service catalogue portal. The NEANIAS portal will offer browsing and keyword search functionality to enable users to discover and find services in an intuitive manner. The service catalogue and each service entity will be described by a clear set of metadata (schema), the Service Description Template (SDT) provided by the EOSC Portal onboarding team, and more specifically the EOSC Enhance project which is responsible for operating and implementing the current phase of EOSC portal. The data catalogue will be provided through

[2] https://eosc-edmi.github.io/

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

OpenAIRE's Zenodo platform (https://zenodo.org/). Zenodo's metadata is compliant with DataCite's Metadata Schema (https://schema.datacite.org/) minimum and recommended terms, with a few additional enrichements. Metadata of each record is indexed and searchable directly in Zenodo's search engine. Metadata of each record is sent to DataCite servers during DOI registration and indexed there. Both the service and data catalogues will support versioning. A globally unique and persistent ID (PID) will be assigned to every resource via Zenodo.

Accessible data. The NEANIAS portal will provide continuous access to the service and data catalogues via the web. The NEANIAS service catalogue will also provide continuous access to the catalogue via standard APIs that are HTTP REST APIs. The description of the Service Model is documented in the API documentation page and is available for download in JSON formats (https://github.com/eInfraCentral/docs). The data catalogue will be provided through OpenAIRE's Zenodo platform. All metadata in Zenodo for individual records as well as record collections is harvestable using the OAI-PMH protocol by the record identifier and the collection name. Metadata is also retrievable through the public REST API. Data and metadata in Zenodo will be retained for the lifetime of the repository.

Interoperable data. Each service collected in the NEANIAS service catalogue will be stored in the Service Data Model as a separate XML/JSON file. For metadata interoperability, service schema will reuse terms from widely known vocabularies and ontologies such as the Dublin Core terms and Simple Knowledge Organization System (SKOS), including taxonomies and classifications for enumerated attributes of a service. Similarly, Zenodo uses JSON Schema as internal representation of metadata and offers export to other popular formats such as Dublin Core or MARCXML. Interoperability will be supported by the provision of dedicated APIs that allow the import and export of the content in standard formats (XML, JSON) and standard data models.

Re-usable data. The public content within the NEANIAS catalogue and services will be available for download and re-use with no restrictions or embargo. The content will be available under permissive licenses, (CC-BY 4.0, CC-0 or comparable) but certain conditions (e.g. Noncommercial use=NC) and/or exceptions may also apply. The NEANIAS service catalogue will be built based on the eInfraCentral software which is available under the GPL/A license. In the cases where specific service information cannot be publicly shared, the reasons will be mentioned in their metadata descriptions (e.g. ethical, rules of personal data, intellectual property, commercial, privacy-related, security-related). In Zenodo, the code is open source, and built on the foundation of the Invenio digital library (https://invenio-software.org/) which is also open source. The work-in-progress, open issues, and roadmap are shared openly in GitHub. All meta data is openly available under CC0 license, and all open content is openly accessible through open APIs. License is one of the mandatory terms in Zenodo's metadata and is referring to an Open Definition license (https://opendefinition.org/). Data downloaded by the users is subject to the license specified in the metadata by the uploader.

2.7.2. C2 core services

Most of C2 Services are infrastructure level services and are not directly handling data at the level to impose data FAIRness policies. C2 services shall be open to operate on all data policies that may be employed by higher level services, including those of selected business cases.

Nevertheless, services provided are able to support the FAIR principles applied by other core services, especially C1 services, as well as C3 and C4 and all thematic services. Mostly related to those are services that allow data depositing, transfer and sharing that will allow carrying out NEANIAS policies related to the preservation and access to data.

2.7.3. C3 core services

As data is the main element in C3 AI services, and data is mostly handled by C1 services, FAIRness is supported indirectly through the policies of C1 core services.

As the result of machine learning algorithms, trained models are produced, which are served (C3.2), and models will also be made accessible for re-use purposes, such as transfer learning.

2.7.4. C4 core services

C4 Visualization services will implement FAIR principles by employing open file formats and the use of metadata, including persistent identifiers, to describe the data thus making data findable and will provide searchable metadata as well. In doing this, standards and recommendations from e.g. OGC and IVOA organizations (see Section 2.6 for more details) and C1 data services will be employed making data accessible for open source tools and services to process the data.

2.8. Operation

The delivery and operation of the NEANIAS services adhere to the recommendations reported in Deliverable D7.1 of Neanias, which stem from well-known and established practices as well as from preliminary results of a survey among the involved partners:

- service management processes follow the FitSM standards family, which aim at achievable IT service management;
- software implementation is supported by tools available to NEANIAS service developers and which allow automatic software building, testing and deployment, possibly using Continuous Integration and Delivery (CI/CD) workflows;
- software documentation has a single access point for all NEANIAS services and relies on the popular Read The Docs (readthedocs.org) document sharing environment;
- software, where applicable, should use free and open source licenses and should be published to online code sharing platforms (such as GitHub);
- operational service-related issues and feedback from both NEANIAS internal sources and external users will be collected through specific tools, already deployed as part of the activities of task T7.2;
- operational services will be monitored, the related quality metrics will be collected and, where applicable, automatically reported to the EOSC facilities;
- the development, delivery and operation of NEANIAS services can rely on the infrastructures operated by consortium members;

- paradigms such as high availability and Infrastructure as Code (IaC) should be adopted when designing, developing and operating the NEANIAS services.

2.9. Security

Security in NEANIAS is based of 3 elements:

- The **user**: the user is defined, authenticated, informed and granted various forms of access and has a central role in the definition and enforcement of security. Abstractions such as “groups” or “roles” may be present still referring to the user.
- The **service**: the service has the main responsibility of applying a security policy to resources that it manages (data, cpu, logic, storage etc). This security policy is defined on a per service basis and is not unique per service. Yet NEANIAS project will define a set of policies that services will opt to follow for simplifying the interpretation of each service commitment.
- The **infrastructure**, that offers the glue among users and services. It translates, in a trustworthy manner, users, roles, groups into elements that can be consumed by services in order to apply their policies. It also supplies support to link users to those policies on a per resource basis.

2.9.1. NEANIAS AAI Concepts

In this section the infrastructure perspective is presented, and through this the representation of users and the instruments offered to services for application of their respective policies.

With respect to security, NEANIAS will offer a horizontal solution that can be utilized by all the underlying NEANIAS services. This solution will focus on the aspects of Authentication and Authorization. Another important aspect that the security considerations can propose and facilitate alternative approaches is that of request delegation.

With respect to authentication, the identity federation paradigm (https://openid.net/specs/openid-connect-federation-1_0.html#rfc.section.1) will be used to facilitate the authentication and registration to the NEANIAS catalog of users, principals that are authenticated through external identity providers. Comprising the identity federation, the following cases can be identified:

- NEANIAS Single Sign On (SSO) – Users registered through the NEANIAS Consortium Single Sign On solution can be authenticated through the respective identity provider and are part of the user federation (https://sso.neanias.eu/)
- EOSC AAI – Users authenticated through the European open Science Cloud Authentication & Authorization Infrastructure (EOSC AAI) (https://eosc-portal.eu/) will also be authenticated to access the NEANIAS infrastructure. Through this identity federation scheme, users with access to a wide variety of identity providers gain access to the NEANIAS services. Some of these providers include:
    - EDUGain Access Check
    - EGI Check-In
    - B2Access
    - OpenAIRE
    - ORCID

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- OpenID Connect Identity Providers – The authorization solution employed by NEANIAS is based on the OpenID Connect protocol (OIDC) (https://openid.net/specs/openid-connect-core-1_0.html). Identity providers that expose OIDC compatible endpoints can be registered and used to authenticate users
- Social platforms – In addition to all other identity providers, if required, widely used social providers such as Google, Microsoft, Facebook, Linked In, etc could be used to authenticate users
- Local users – Beyond external identity providers, the option to authenticate users through local user accounts will also be available. It is expected that this method will primarily be used for maintenance and administration accounts

With respect to the supported grant flow through which the requestor identity is federated and subsequently propagated to servicing endpoints, the following mechanisms can be identified:

- Authorization Code Grant Flow (https://tools.ietf.org/html/rfc6749#page-24) – This flow is usually the preferred method to authenticate users via Open Id Connect. In the first request, the user is redirected to the external identity provider where he is authenticated and then redirected back to the identity consumer site with an authorization code. Subsequently, another request is made to exchange the authorization code with a token set containing information such as access, refresh and id token. This flow will be utilized to authenticate users through external identity providers.
- Client Credentials Grant Flow – This flow can be used for machine to machine authentication. In this grant, a specific user is not authorized but rather the credentials are verified and a generic access token is returned. This flow will be utilized to authenticate NEANIAS services in order to interoperate with other NEANIAS services
- API Keys – This method is not part of the Open ID Connect specification and will not be centrally managed. It is rather an option for services that will require a simpler approach to authenticate client requests, whether these are within the NEANIAS ecosystem, or to external services.

The authentication flow that passes through the centrally managed NEANIAS AAI service will expose suitable user info endpoints (https://openid.net/specs/openid-connect-core-1_0.html#UserInfo) through which the respective id tokens can be used to retrieve the caller's claims. This information will be provided in the form of a JWT token (JSON Web Token) (https://tools.ietf.org/html/rfc7519) containing information to be utilized by servicing components.

With respect to authorization, the following alternatives can be made available and can be utilized based on specific service needs. It should be noted that these alternatives are not necessarily mutually exclusive, and a combination can be used by some service as seems most fitting.

- Role based – The mechanisms to define roles within the NEANIAS AAI Single Sign On (SSO) solution will be provided. These roles can be subsequently assigned to authenticated users and the information flow down to services with each request as

the respective caller claims. The roles can be defined with a global application scope, or they can be client specific. This way, each service can define a set of roles that do not clash in semantics with other service needs. In addition to roles, user groups can be employed and semantics on group membership can be utilized by the services or even be used as an administration utility to easily propagate affective roles to groups of users

- Resource based – A mechanism to define ad hoc resources within the NEANIAS AAI service can be investigated. The purpose of such a construct would be to allow NEANIAS services, through respective exposed API endpoints to autonomously define resources and manage user authorization over them. The scope of these resources can be determined in an ad hoc fasion by each service, although it is expected that coarse grain resources would be used
- Service Specific – it is expected that each service will have diverse authorization requirements. For this reason, in addition to the Role and Resource based authorization that could be offered horizontally, each service could employ fine grained authorization policies within its own business logic. For this case, the horizontal NEANIAS solution would be limited to providing consistent client subject identifiers and any origin information it has available to facilitate in-service solutions, possibly, ranging from access control lists (ACL) to custom business rules

With respect to request delegation, the approach taken can be differentiated depending on the caller and recipient requirements. The initial recipient receiving a user request will have all the tracking and identification information of the caller through the respective JWT token. In case this recipient requires, in the flow of normal execution, to invoke other services to complete or forward the service request, two cases can be distinguished:

- Service Identification – The underpinning service may only require that the caller service is authenticated, and any authorization checks are handled at the level of service requestor. Tracking and accounting actions are handled by the recipient at the level of caller service and further bindings with respect to the request originator is accounted by the caller
- User Identification – The underpinning service requires complete knowledge of the request originator to handle aspects of its operation such as accounting, authorization, and any other business logic rules.

Depending on the use case, different alternatives can be examined.

- Instead of requesting a new access token for a service to service authentication, services that need to delegate user information to other services can use the access token send from the user. In this case the initial access token requested by the user, should have access to all the scopes required to finish the initial as well as subsequent request
- When performing service to service communication, apart from the access token services can also include in the API invocation enough information on the user that they are acting on behalf of. This can be in the form of a direct API parameter or even through some request header. The receiving service must be aware of this header and able to propagate it to potential subsequent requests

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

In the latter case caution needs to be applied to make sure that the propagated information is not tampered with over untrusted network and unsecure communication channels. Countermeasures against this could include trusted networks, transfer level encryption as well as signing the propagated information through well-established standards (public-private key encryption and signing). To facilitate the full lifecycle of service development, the security environment can be sandboxed to differentiate between production, staging and development environment. Different users and federation identity providers can be supported in each of the environment specific sandboxes.

2.9.2. Policy

Security policy is applied on a per service basis. However, the following general policy shall be applied in NEANIAS:

- Each service presents a terms of use document where its precise policy regarding security is presented.
- Each service may opt to pick one of predefined data security policies that will be presented by the project in subsequent stages.
- Services should clearly present to the user any risks for exposing data uploaded or utilized by the service to 3rd parties, defining those 3rd parties.
- Services should prevent unauthenticated access to their end-points, unless those endpoints are intended for public use, in which case the endpoints should employ throttling or another protective mechanism to prevent misuse of infrastructure resources and should avoid consuming substantial network, storage or network resources.
- Services should apply some mechanism to prevent unauthorized data access. The granularity this is applied should be evident to the provider of data.
- All services must avoid placing sensitive or otherwise protected data in their logs and accounting records, if this is not strictly required for performing their activity. Any such data use should be made evident to their user and get her consent.
- Services should avoid profiling of users unless necessary. In such case they should present the user with relevant information and need and get her consent.
- Services shall preserve user consent statements according to GDPR guidelines.
- Services shall preserve any sensitive information according to GDPR guidelines.

2.10. User Interfaces / User Experience

NEANIAS portal and services will be offering web-based graphical user interfaces to allow users to perform various actions. These interfaces will be designed based on the basic principles of web design in order to offer users a frictionless and engaging user experience.

These basic principles include the following guidelines:

Usability: any design element of the user interface should have a purpose. A less-is-more approach could help emphasize simplicity as opposed to clutter.

Simplicity: simple language that is clear and concise should be used as it can be easily understood by different types of users and helps reduce ambiguity at the same time.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Consistency: the user interface should be consistent by making the visual presentation of the same type of information and behavior of the same interactive elements predictable, thus ensuring continuity of previously acquired knowledge.

Narrativity: users should be able to understand the relationship between specific actions and why they happen, what happens afterwards, and how that affects future events, all based on the flow of time.

Feedback: the user interface should have an interactive design that responds to user actions and encourages communication. For example, a clicked icon could change color or shape.

Intuitive and user-friendly interface: a friendly user interface should allow only for an appropriate set of actions and prevent situations that result in errors as for example warn users for actions where they might damage data. At the same time the use of familiar and recognizable concepts widely understood by all types of users could help users navigate more easily without having to follow a learning process.

Compatibility: the user interface should make use of technology widely available to users.

Responsiveness: the web design should be responsive in order to render well on a variety of devices and window or screen sizes used by different users.

Accessibility (optional): the user interface design should be usable by as many people as possible including people with disabilities.

Although NEANIAS user interfaces will be implemented by different service providers using a variety of technologies, a general NEANIAS web toolkit will be offered with optional elements and guidelines to enable all services' user interfaces to have a similar and recognizable look and feel. This toolkit will include coloring guidelines, texts, html elements, proposals for menus, generic icons, possibly dynamic widgets and is analyzed in more detail in section 5.5.

2.11. Portability

NEANIAS service portability approach attempts to capture the existing landscape of services that are initially onboarded the project, their targeted evolution and the constraints that may underly those.

Implementation-wise, NEANIAS strongly suggests that services are implemented adopting portable technologies and rely on open standards for their interactions and dependencies so that they can be ported to different infrastructures. Furthermore, NEANIAS suggests that services are packaged in forms that can be ported across cloud infrastructures avoiding vendor lock-in. Examples of portable implementation technologies are java, python, .net core etc. while portable packaging technologies are war, jar, docker images etc.

Yet, service portability in NEANIAS may follow any of the following approaches, assuming that there are solid justifications to deviate from the optimal maximum portability case:

- **Portable Service**: A service that can be deployed in any major cloud infrastructure with minimal assumptions. Services may have preferences due to data/service locality exploitation patterns, however those are not technically limiting the deployment of the service in any other infrastructure that satisfies its limited restrictions of requirements.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- **Bound Service**: A service that is bound to an infrastructure or technology and needs to be deployed to it due to a number of reasons such as fixed local dependencies, security etc. (e.g. a service that requires internal access to a long-term storage tape facility, a specific version or flavor of operating system). Although bound to the infrastructure or technology the service is still a dynamic in terms of instantiation.
- **Fixed Service**: A fixed service is the most limited type of service as it is tightly bound with an infrastructure and/or technology and is expected to be a fixed point of reference with substantial static features. For instance, the servicing API of a large datastore is such a service, tightly bound to the technology and the physical resources that underlie the service.

Admittedly the boundaries among the three classes of service may not be sharp, however the classification will greatly help service providers shape their service provisioning pattern.

2.12. Technology

The following technologies, frameworks and systems are used for the implementation of NEANIAS Core Services:

2.12.1. Programming languages

- Java: A general-purpose, object-oriented and strongly typed programming language. Along with other technologies that compose the Java Platform (e.g. the Java Virtual Machine), it is widely adopted for cross platform service implementation.
- C#: A multi-paradigm, object-oriented and strongly typed language developed by Microsoft. It is used worldwide for implementing desktop applications, web applications and web services.
- Python: An interpreted, loosely typed, general-purpose programming language. Its constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
- JavaScript: An interpreted, prototype-based programming language specifically optimized for developing client applications.

2.12.2. Development frameworks

- .Net Core: A free open source, state-of-the-art framework for portable services implementation that builds upon the Microsoft .NET framework, enabling several computing languages, including C#. It will be used for the implementation of several Core services.
- Angular: A widely adopted state of the art framework for cross browser web applications delivered under the SPA paradigm, building on the MVC design pattern for empowering structured, rich, client applications.
- NodeJS: An asynchronous event-driven JavaScript runtime, designed to build scalable network applications.
- Jupyter: An open-source project to support interactive data science and scientific computing across all programming languages.

2.12.3. Data Management frameworks

- ADAM: The Advanced geospatial Data Management platform is a tool to access a large variety and volume of global environmental data. ADAM allows you extracting global as well as local data, from the past, current time, as well as short term forecast and long-term projections. Most of the data are updated daily to allow users having always the most recent data to play with.
- Rasdaman: A data management and analytics system for massive multi-dimensional arrays ("datacubes") such as sensor, image, simulation, and statistics data appearing in domains like Earth, Space, and Life sciences; in particular its main features include flexibility, performance, scalability, and open standards support.
- GeoServer: open source server for managing, visualizing and sharing geospatial data.
- PostgreSQL: a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.
- CKAN: an open source metadata and basic data management system to support data registration, discovery and sharing.

2.12.4. AI/ML frameworks

- Tensorflow: An open-source platform for machine learning; it provides a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML.
- Keras: A high-level, open-source, neural networks API, written in Python and capable of running on top of different back-end platforms, among which Tensorflow; it is designed with a focus on enabling fast experimentation.
- Spark Mlib: a distributed machine-learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is faster than most disk-based implementations.
- DeepLearning4J: Eclipse Deeplearning4j is the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. Integrated with Hadoop and Apache Spark, DL4J brings AI to business environments for use on distributed GPUs and CPUs.
- Streamlit.io: an open-source app framework for Machine Learning and Data Science teams able to quickly turn python code into a dynamic web app.

2.12.5. Visualization Frameworks

- Dash: a productive Python framework for building web applications.
- Plotly: an interactive, open-source plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.
- VTK: open-source, freely available software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and 2D plotting. It supports a wide variety of visualization algorithms and advanced modeling techniques, and it takes advantage of both threaded and distributed memory parallel processing for speed and scalability, respectively.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

2.12.6. Major Background Systems

The following systems are utilized in the background as technologies in NEANIAS Core Services:

- OpenStack: a free open standard cloud computing platform, mostly deployed as infrastructure-as-a-service (IaaS) in both public and private clouds where virtual servers and other resources are made available to users. The platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage, and networking resources throughout a data center. Users either manage it through a web-based dashboard, through command-line tools, or through RESTful web services.
- Kubernetes: an open-source container-orchestration system for automating application deployment, scaling, and management.
- ELK Stack: a system including three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.
- Docker: a popular Linux-based containerization system, which allows to ship applications already bundled with their runtime dependencies.
- Apache Spark: An open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. In particular, it supports Spark MLib, a distributed machine-learning framework that will be employed by AI services.
- Keycloak: an open source Identity and Access Management solution. It provides support for User Identity Federation and Single Sign On scenarios through widely used protocols such as Open ID Connect 1.0 and SAML 2.0.

3. System Architecture

In this section of the report the overall architectural perspective of NEANIAS is presented, in order to allow the reader to comprehend the full breadth and depth of services and capabilities to be offered by the arising services ecosystem.

At first, a diagrammatic overview of the ecosystem is given, in the form of logical layered architecture. Although the precise architectural approach is a mix of microservices and service layers, it is assumed that the layered visualization allows easier visualization of the concepts. To support the reader, brief descriptions of the elements presented in the diagrams are provided. Complementing the architecture, a reference service lifecycle is presented, so that the notion of a "NEANIAS service" emerges, as a piece of software that loosely conforms to some principles.

Through those two major diagrammatic and textual descriptions, one can comprehend the application of various principles of SOA, Open Science and EOSC architecture in general. One can also envisage how core services are brought together to yield reusable generic services and empower the thematic services delivered by the research sectors.

Finally, an indicative hypothetical physical architecture is introduced, in order to present concrete deployment options allowed by the presented architecture and its principles.

3.1. Architecture overview

NEANIAS architecture follows the distributed, n-tier architecture model. In this architecture NEANIAS Services are logical architecture blocks that cover specific functional or technological requirements of the infrastructure and are exposed to other NEANIAS or 3rd parties' services via their contracts. NEANIAS services themselves may, and usually are, composite multi-tier systems encapsulating microservices and protocols for their internal operation. In few cases those internal components may be robust, well defined and exposable on their own into the infrastructure, however they will not be presented as essential building block of the overall architecture presented here, so as to allow the document to focus on the notion of NEANIAS ecosystem of services.

NEANIAS services fall under 8 major groups, which can be logically clustered in 3 clusters

- **Base Core Services**, a cluster of generic services that provide basic tooling for NEANIAS ecosystem services.
    - **Open Science Lifecycle support services (C1)**
    - **EOSC hub Research infrastructures and cloud integration enabling services (C2)**
- **Advanced Core Services**, a cluster of services that build on base ones to deliver higher level generic yet focused in a specific domain services to be exploited by thematic services, which include:
    - **Artificial intelligence Services (C3)**
    - **Visualisation Services (C4)**
- **Thematic Services**, a cluster of services that includes top-level services for:
    - **Underwater research sector services** (Ux), emerging from the requirements and work of WP2.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- **Atmospheric research sector services** (Ax), emerging from the requirements and work of WP3.
- **Space research sector services** (Sx), emerging from the requirements and work of WP4.
- **Business innovation sector Services**, i.e. services that may emerge to fulfill the business sector needs, which remain to be identified and designed in next stages of the project.

Although not strict, the diagram in Figure 1 coarsely depicts the stack of those service clusters and groups in NEANIAS.

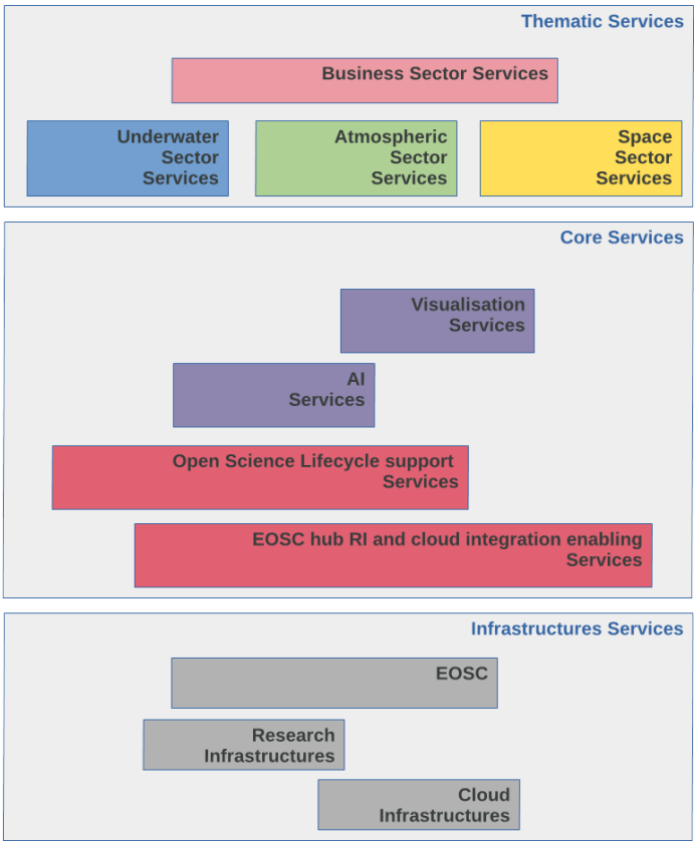


Figure 1: Logical layering of NEANIAS service clusters

3.2. Logical architecture

Figure 2 depicts the coarse logical architecture of NEANIAS.

In this diagram 5 layers of abstractions are depicted:

- Level 0: Physical resources of the infrastructure, such as compute, store, network, users etc
- Level 1: Abstractions of physical resources, such as virtual machines, standard storage APIs, user identities etc
- Level 2: Services that act close to resource abstractions and mainly orchestrate access to those resources adding higher level features.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- Level 3: Services that provide facilities that may reach up to the end user offering functional features, however they are mainly intended for composition of higher-level services rather than standalone ones.
- Level 4: Services that are targeting specific use cases. Might form applications (i.e. sets of services, usually with rich user interfaces that collectively serve some use) or standalones services that provide specific tools to their users.

Apart from those services, there are two sets of services that reside outside or at the boundaries of NEANIAS infrastructure:

- Boundary services, that are defined by NEANIAS, support the provisioning of NEANIAS services but may be residing on or be integrated in external infrastructures to offer their features.
- External services, that are provided by infrastructures not controlled by NEANIAS on their own terms and specifications.

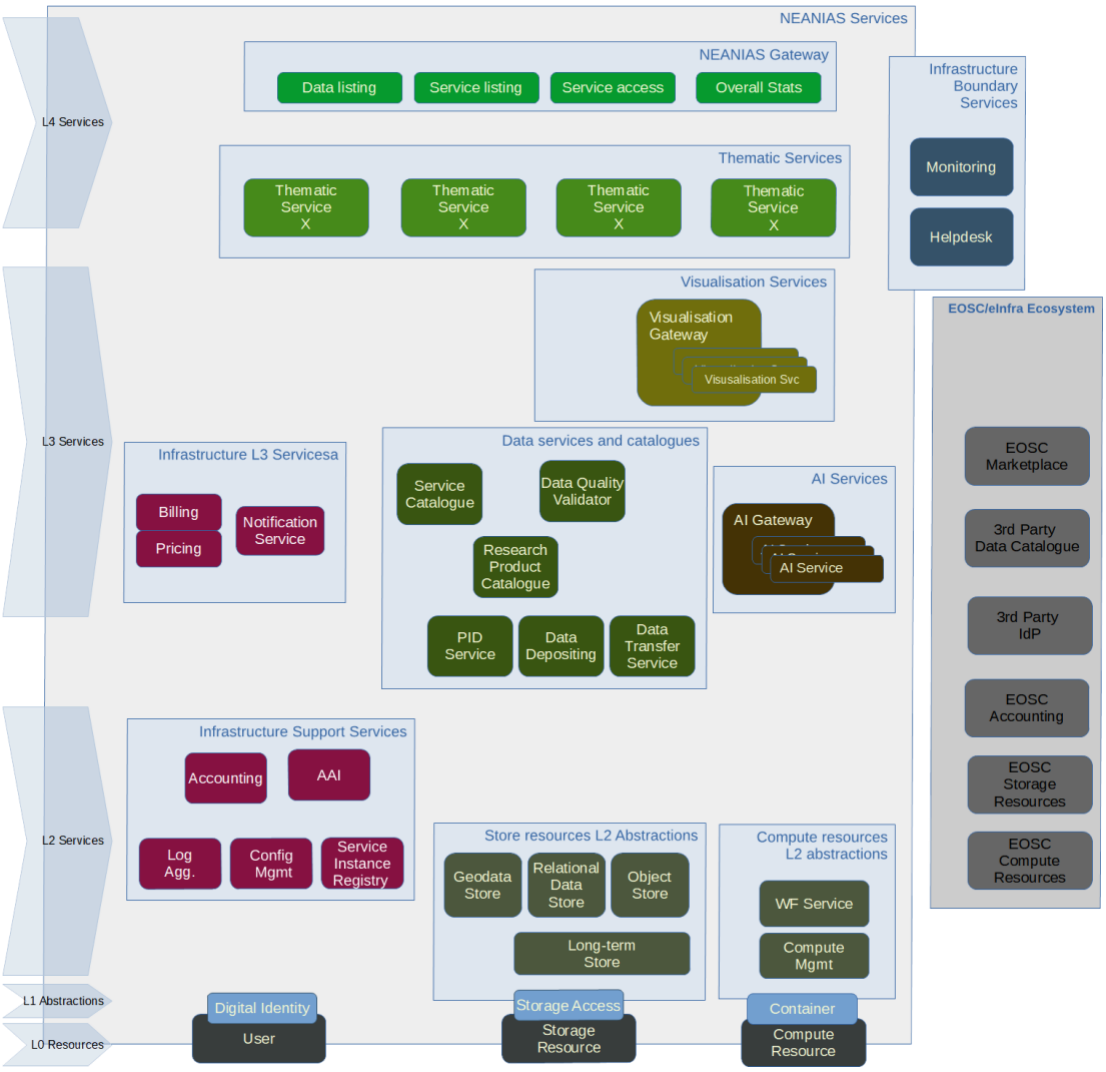


Figure 2: Logical architecture diagram

3.3. Fundamental building blocks

In the logical architecture presented in Figure 2, one can identify the following fundamental elements that support the definition of NEANIAS services:

- **NEANIAS Gateway**: The area where the user meets the NEANIAS service and data offerings. It offers general information on NEANIAS services as well as view of the service and data catalogue and access points to user-oriented services.
- **AuthN/AuthZ** (AAI): The service is a gateway to Identity providers that will be linked to NEANIAS and will allow the authentication of local users. It also supports service authentication and provides fundamental instruments to support client authorization, yet NEANIAS services may opt for different means for the latter.
- **Digital identity**: is the representation of a user in the virtual landscape.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

- **Service Catalogue**: A human accessible catalogue of service offerings of NEANIAS, compliant with the EOSC hub/marketplace directives, where users may locate project's offerings.
- **Service Instance Registry**: A machine accessible catalogue to register and locate active service endpoints via their metadata, essential for supporting a dynamic SOA ecosystem.
- **Research Product Catalogue (aka Data Catalogue)**: A metadata repository listing datasets of NEANIAS. Services may publish and retrieve dataset descriptions from the catalogue, and locate the endpoints where those datasets reside. Federation is supported via well-known protocols.
- **Data Depositing Service (aka Data Repository)**: A data storage abstraction a store to deposit data for later use. Data may be mostly deposited in object forms which in cases may even be not directly usable by consumer services, and response times may be inferior compared to a to data type-specific store.
- **Data-type specific stores**: Datastores specialized to manage data via services that comprehend the internal structure of the datasets. Such may be relational databases, geospatial data stores etc.
- **Configuration Management**: a distributed redundant service for storage of configuration information and secrets required by services instances to perform their operations.
- **Long Term Store**: a backend storage abstraction that can be trusted for long term preservation of datasets. The service may be even offline, resulting in response times substantially inferior compared to other store technologies.
- **AI Service**: an umbrella service that encloses a number of technologies for Artificial Intelligence related tasks execution, which start from selection of algorithms, cover algorithm training and subsequently application in various tasks (e.g. prognosis) and feedback integration. The service will encapsulate training datasets.
- **Visualisation Service**: an umbrella service that encloses a number of technologies for visualizing and exploring specific data types. It may be coupled with specific clients that deliver the visualization the end-user
- **Compute Management**: An abstraction for management of infrastructure containers that may apply policies for optimizing the performance and resilience of the system, if services and infrastructure allow it. It provides specific CPU and memory abstractions.
- **Container**: The main carrier of service executable elements, packaged and configured by service publisher. At least one is being defined by NEANIAS, but more may emerge in the course of the project, if compliant with infrastructures available.
- **EOSC AAI**: The AAI of EOSC which will be one of the main IdPs engaged by NEANIAS AAI.
- **EOSC Compute**: An abstraction for computation offered by EOSC service marketplace, that may be utilized on demand by NEANIAS services. EGI compute may be one of those.
- **EOSC Storage**: An abstraction for storage offered by EOSC, that may be utilized on demand by NEANIAS services. EUDAT B2Store is one example of such storage service.

- **3rd Party Data Catalogue**: Other catalogues, such as OpenAIRE, Zenodo etc that may be linked to NEANIAS data catalogue, to draw dataset descriptions for further promotion of open science.
- **EOSC marketplace**: The area where all NEANIAS services are listed for discovery by researchers and scientists.

3.4. NEANIAS reference service

In Figure 3 an indicative lifecycle of a service starting up to serve its users in NEANIAS is presented. In this flow, the following steps may be present:

- Initialization: any work performed by the service before starting interaction with NEANIAS services. May refer to container handling, point 0 configuration application etc.
- Configuration discovery / access: the service discovers the configuration options needed for its operation. Those might be drawn from local store (i.e. container defined) or from infrastructure services. Configuration may refer to default operation parameters, fixed dependency endpoints, secrets etc.
- Service dependency discovery: the service may access the catalogue or other technologies in order to discover the services it will depend on, in order to provide its features to its users. In this stage the access points of services that satisfy the requirements of the service are located.
- Data discovery: the service may discover datasets that might be fundamental for its operation. Examples might be a training set of an AI service, or map terrain data for a mapping service.
- Store allocation: in case the service will need pre-allocated store, at this point it checks to acquire it.
- Compute allocation: in case the service will need some allocated computational power or memory size, it could ask for it before staring up.
- Data fetch: following store allocation, the service might request the transfer of data it needs to use in the allocated storage.
- Process: when prerequisite data and relevant resources are ensured, the service may process those in order to initiate its servicing lifecycle. Processing may range from simple validation to calculation of complex structures that will allow the service to respond to its client's requests.
- Registration: At this point the service is ready to start responding to requests from clients, thus it registers itself to the instance registry, for others to be able to discover it.
- Listening / monitoring: The service is listening to client servicing requests, management requests (if supported) and monitoring requests.
- Un-registration: As a respond to a specific request posed by the hosting infrastructure or a privileged caller, the service unregisters itself
- Cleanup: The service cleans up any allocations it might have performed.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

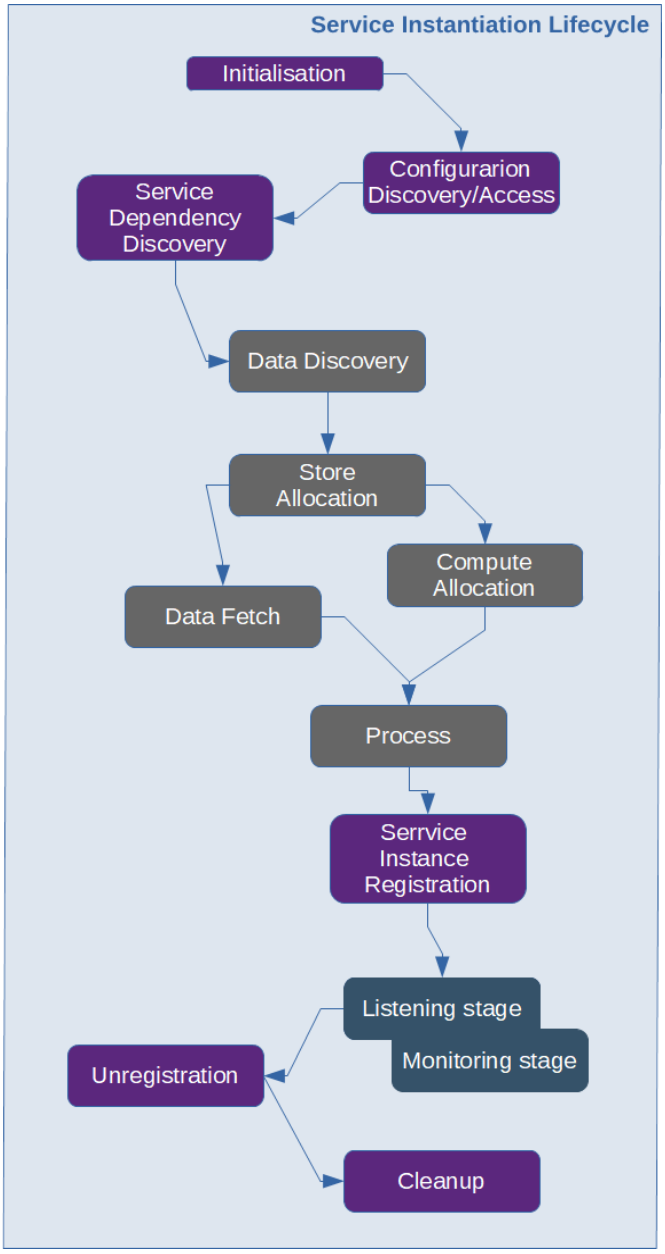


Figure 3: Reference service instantiation lifecycle

In Figure 4 an indicative lifecycle of a service serving a client request in NEANIAS is presented. There, one can identify the following steps:

- The client invokes the service
- Authentication of the client is performed under one of the supported models.
- Initial authorization is performed, which filters out whether the specific client may be invoking the particular endpoint, potentially analyzing the parameters of the

invocation in case a resource-dependent authorization model is supported by the service (e.g. ACLs)

- In case additional configuration needs to be discovered at this point the service accesses either local or central configuration. One example could be the access to specific secrets due to specific users invoking the service etc.
- Late service discovery may take place at this point for services that may need to react to a fast-changing environment. However, this should take into account that the active discovery cost needs to be justified to the caller.
- Depending on the request, the service may need to locate and access specific data. Those may be explicitly or implicitly defined by the caller, or the service may be acting on dynamically located data that need to be sought for.
- The service may need storage to move discovered data or data to be generated. Such storage may need to be allocated, or ensured at this point, before data fetch or generation is initiated.
- If data need to be fetched close to the service, then those are transferred at this point. The process may be an asynchronous or incremental one, changing substantially the service lifecycle, according to its design needs.
- Similarly, the service may depend on computational power or memory that need to be allocated in the local or other infrastructures.
- Once all resources are present the service may invoke a processing cycle which may generate new data or prepare existing data for exploration.
- Data generation although not technically separate from processing is a “semantic stage” when new data are generated by a process.
- Persistence may follow data generation stage, if those data may serve further uses. Persistence may take place in mid/long term service storage areas of the project, i.e. escape the local storage of a virtual node.
- Exploration of data, via a service specific approach, may follow any of the previous optional stages, i.e. processing, generation or persistence.
- Last step may be the publication of data to catalogues and public data repositories.

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

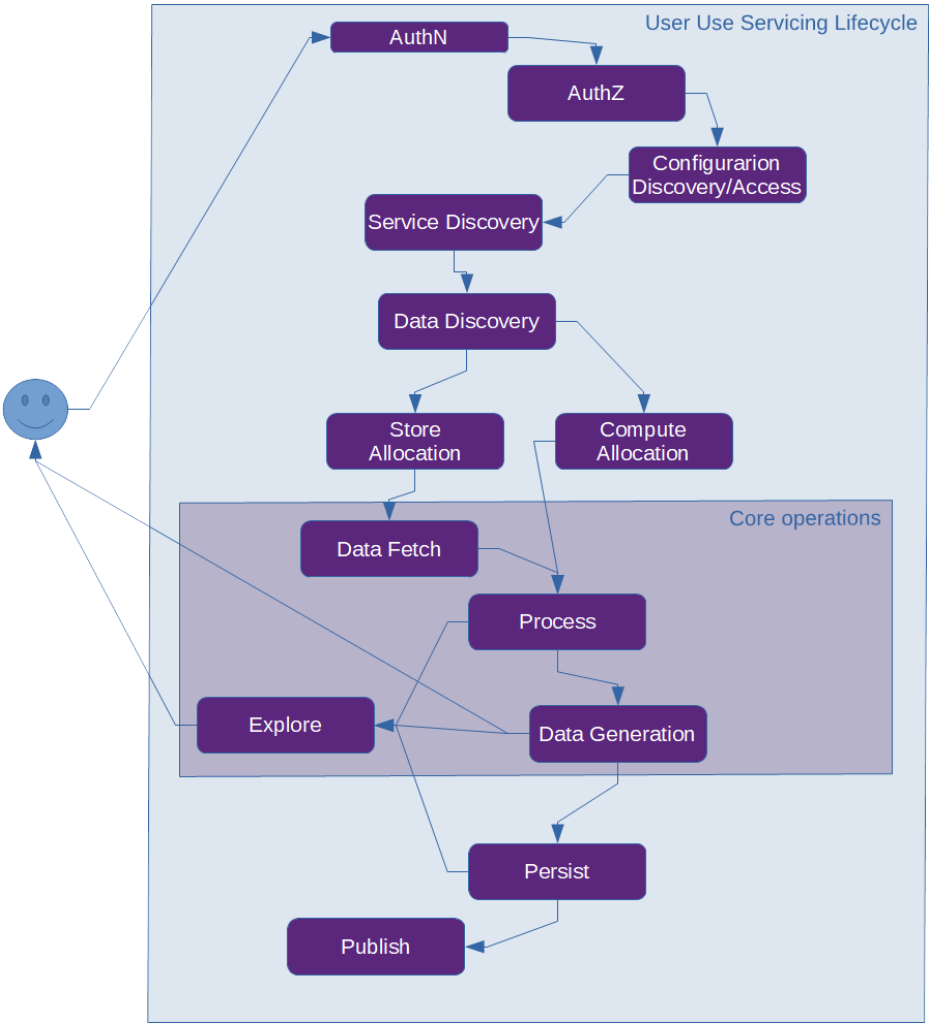


Figure 4: Reference servicing lifecycle

3.5. Physical architecture

In Figure 5, a potential deployment scenario for NEANIAS services is presented.

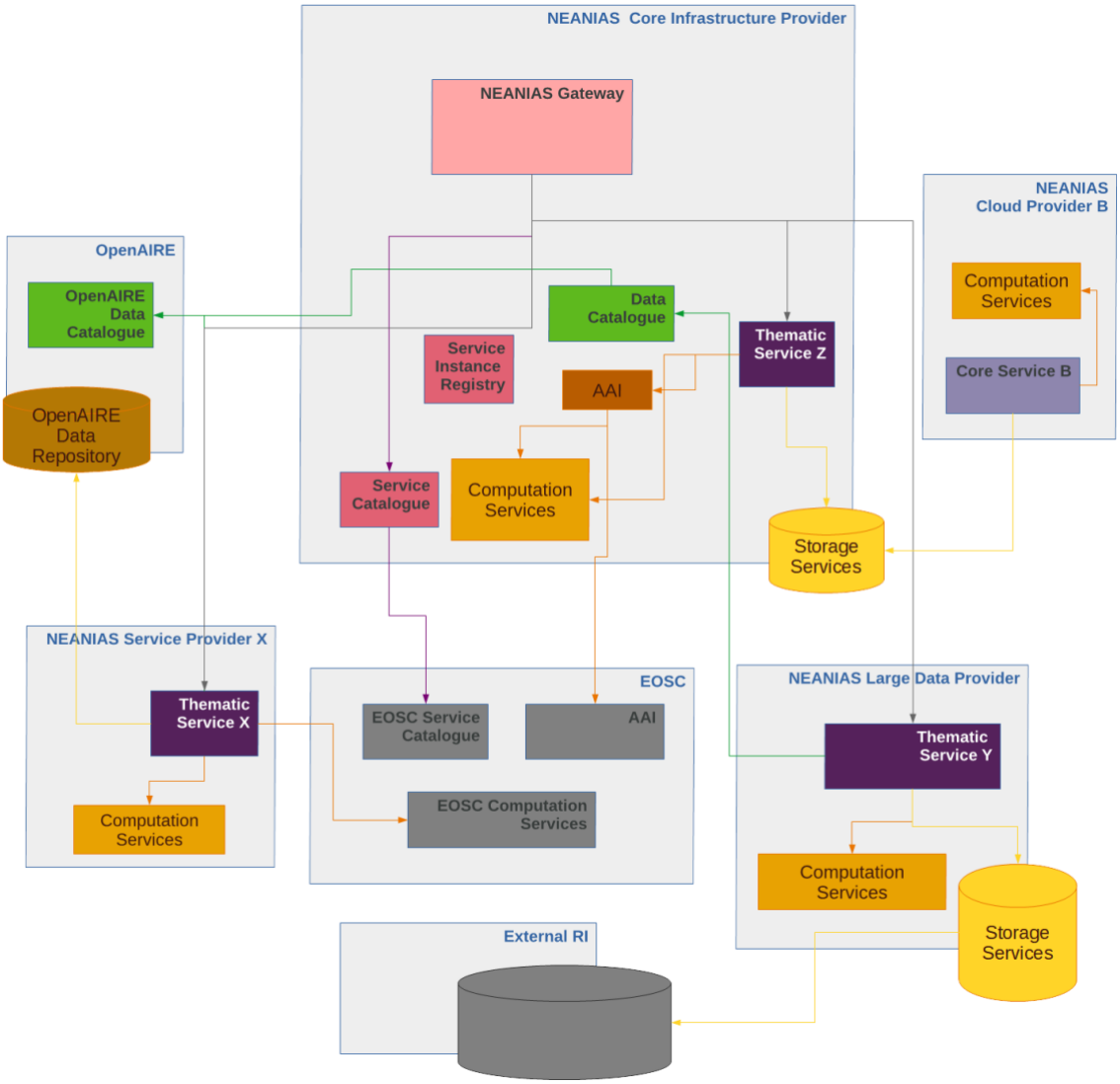


Figure 5: Hypothetical deployment of NEANIAS services

In the presented physical architecture scenario, the following assumptions are made:

1. A core infrastructure provider for NEANIAS is present, hosting services:
    a. NEANIAS Gateway
    b. NEANIAS AAI
    c. Service Instance Registry
    d. Data Catalogue
    e. Service Catalogue
    f. Storage Services (group of services)
    g. Computation Services (group of services)
    h. A portable Thematic Service Z that depends on storage and computation offered by the infrastructure provider.
2. OpenAIRE is present with two services.

   a. The OpenAIRE data catalogue, that harvests metadata from NEANIAS Data Catalogue, following the validation of the service and the compliance with OAI-PMH protocol.
   b. The OpenAIRE data repository that is hosting data (and metadata) pushed to it by services, which is directly utilized by Thematic Service X of the scenario.
3. EOSC is present with 3 services:
   a. The EOSC Service Catalogue, which includes information from NEANIAS Service Catalogue
   b. The EOSC Computation Services (offered by a 3rd party) that is used by Thematic Service X
   c. EOSC AAI service that is linked to NEANIAS AAI.
4. A NEANIAS Large Data Provider, which hosts a large data volume that may be infeasible to be moved for practical reasons, or a data volume that is locked behind access policies, that hosts:
   a. Local storage exposed via protocols that match the data type managed and facilities offered on top of those. Those data themselves may be well a portion or replica of data present in another infrastructure (External RI C).
   b. Local computation services, that allow computation to run next to data.
   c. Thematic Service Y, which is a service bound to the data and computation services of NEANIAS Large Data Provider.
5. An External Research Infrastructure C that hosts primary or replicas of primary observation / modeling data, which usually consists of petabytes of digital objects in sector specific formats.
6. A NEANIAS X service provider, that offers
   a. minimal local computational capacity to instantiate its service X
   b. Thematic Service X, which is mostly relying on external EOSC provided computational resources (e.g. EGI) to deliver is computation bound results and data storage services provided by OpenAIRE Data Repository.
7. A NEANIAS Cloud Provider B that hosts
   a. Core Service B, which depends on data managed by the NEANIAS Core Infrastructure Provider.
   b. Local computation to allow execution of Core Service B.

4. Fundamental resource abstractions in NEANIAS

4.1. Storage

The MTA SZTAKI Cloud, introduced in D7.1 of Neanias, based cloud environment[3]The MTA Cloud provides storage abstraction via the OpenStack Cinder API. Cinder provides volumes, which are raw unformatted virtual block devices that can be attached to a virtual machine instance. These can then be used as a traditional hard drive by the instance's operating system. Cinder also provides Snapshot functionality. These snapshots are a read-only, point-in-time copy of a volume's contents. A snapshot can be created from a volume that is currently in use or in an available state. The snapshot can be used to create a new volume, too. Cinder also provides backup functionality to create compressed, archived file of a volume's contents and store in a Swift object storage container, or at another third-party object store provider. These volumes rely on a distributed Ceph cluster (https://docs.ceph.com) in the background. Operating System images, from which Virtual Machines can be created, are instead managed by the Glance OpenStack component, from which Virtual Machines can be created, are instead managed by the Glance OpenStack component.

The GARR Cloud Platform, also described in Deliverable D7.1 of Neanias , and thus provides the same storage abstractions as the MTA Cloud. In addition, the GARR Cloud Platform provides an S3 compatible RESTful API to manage objects and object containers. Although not exposed to the end users, all the storage in the GARR Cloud Platform is provided through Ceph, which abstracts the hardware layer and implements data redundancy.

The GARR Container Platform is instead based on Kubernetes, a container orchestration PaaS system. Kubernetes volumes, which in the GARR Container Platform are provided by Ceph, are used for data persistence.

4.2. Computation

The MTA Cloud and the GARR Cloud Platform provide computational resources as virtual machines via the OpenStack Nova component. Nova is a collection of daemons that work together to orchestrate availability of compute resources, leveraging virtualization features on compute nodes. Nova works with a variety of existing hypervisor technologies and abstracts available CPU (vCPUs) and memory resources on compute nodes to create “instances”, i.e. virtual machines. To this aim, the MTA Cloud and the GARR Cloud Platform use the QEMU-KVM technology. OpenStack provides pre-defined virtual machines templates to create virtual machines via the Nova API or the Horizon dashboard as well.

Concerning datacentre level abstractions, in the GARR Cloud Platform, resources located in different geographical regions (which correspond to different datacentres) can be employed through the same API endpoints.

[3] https://www.openstack.org/

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

On top of the GARR Cloud Platform, GARR provides DaaS, i.e. Deployment as a Service, which automates the deployment of applications in OpenStack and which is based on Canonical's Juju[4].

In Kubernetes, and thus in the GARR Container Platform, computational resources (including GPUs) are abstracted through "pods", which are groups of Docker containers sharing the same networking stack, and which can be deployed on "workers", i.e. server abstractions. Kubernetes provides also deployment abstractions, which allow to scale and upgrade pods.

Moreover, the GARR Container Platform allows the easy deployment of Jupyter notebooks through Helm.

4.3. Other resources

The MTA Cloud and the GARR Cloud Platform provide SDN (Software Defined Networking) for the virtual machine communication via the OpenStack Neutron component. Neutron is a networking service that manages the OpenStack environment's virtual networks, subnets, IP addresses, routers, firewall rules, and more. Neutron allows users to create the necessary virtual resources not only to ensure that their instances obtain internal IP addresses (also known as fixed IPs), but also to have the ability to map external IP addresses (also known as floating IPs) to instances. This allows applications residing in OpenStack instances to be externally accessible. Neutron also provides firewall functionality for the externally accessible services with Security Groups. With Neutron, users can view their own networks, subnet, firewall rules, routers, and load balancers, all through the Horizon dashboard or the Neutron API.

In OpenStack resources are assigned to *projects*, to which many *users* can be added with a *role* (e.g. Member or Admin).

In Kubernetes, resources are instead grouped into namespaces. Moreover, Kubernetes provides the service abstraction, which in turns relies on other abstractions such as load balancers, deployments and pods.

[4] Juju - How it works: https://jaas.ai/how-it-works

5. C1 Open-Science lifecycle support services & Components reference

C1 services, the Open Science lifecycle support services, enable the integration of NEANIAS thematic services with EOSC hub and offer providers, services and users the ability to publish, present and locate resources.

In these section C1 services are described. First, the role of C1 services is briefly outlined and then each service is presented based on the Service Description template guidelines of the project.

5.1. The role of C1 services in NEANIAS

C1 services will provide the necessary tools for NEANIAS services to be discoverable and accessible and integrated with EOSC hub. The NEANIAS Portal and NEANIAS service catalogue built on eInfraCentral[5], will enable service providers to register and present their services. Through the NEANIAS portal and service catalogue users will be able to browse, search and discover all thematic and core services available. At the same time, the NEANIAS data catalogue will enable providers to publish their data while the NEANIAS PID service will allow the generation of PIDs for digital assets making them unambiguously cited and therefore uniquely discoverable. As part of the NEANIAS C1 services, providers will also be able to validate datasets via automated or human driven processes and create actionable Data Management Plans with the use of the Argos service provided by OpenAIRE.

5.2. NEANIAS Service Catalogue

Short Name	Catalogue Service	**Lead partner**	ATHENA
Type	Web service (REST API)	**Contributors**	-
Title	NEANIAS Catalogue Service		
Master element	Open-Science lifecycle support services implementation		
Description	NEANIAS catalogue service will offer the necessary service catalogue REST APIs for a. NEANIAS service providers to register and update the service metadata in the NEANIAS catalogue of services as well as monitor the usage (e.g., pageviews) regarding their services. b. 3rd party systems including EOSC portal to synchronize the service metadata with the service catalogue of NEANIAS.		
Technical details	Web service built JAVAEE and PostgreSQL offering a list of REST methods for managing a catalogue of EOSC compliant services. Based on the eInfraCentral FOSS https://github.com/eInfraCentral.		
Core integration	It needs the following integration		

[5] https://einfracentral.eu/

	• Authentication / Authorisation for authenticated users to login and update service metadata.		
Depends on	EOSC AAI		
Use cases	a. NEANIAS service providers use APIs to programmatically update the catalogue and also to publish services into EOSC • 3rd party systems may use the APIs to access the content of the catalogue		
EOSC services integration	EOSC AAI EOSC portal, www.eosc-portal.eu		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	NKUA \ ATHENA RC	**License**	Apache License
First availability	2020-08 Planned month of preliminary release to support NEANIAS outreach and engagement plan and brief reference of what may be usable at the point. No significant integration is expected.		
MS4 expectation	Prototype deployment. Not populated with NEANIAS service metadata. EOSC integration is not mandatory for EOSC services yet.		
MS6 expectation	Deployed and populated with metadata of all NEANIAS services.		
MS7 expectation	Issue fixing and improvements.		

5.3. NEANIAS Research Product Catalogue

The NEANIAS Research Product Catalogue (RPC) is the service that will store all products of the NEANIAS project, along with datasets that the users of the NEANIAS services which will make available online.

Research products of the NEANIAS project include:

- Publications
- Software
- Datasets used as input to NEANIAS services
- Datasets produced by the NEANIAS services that the users wish to make available to others

For the research product catalogue, the NEANIAS project will use the OpenAIRE's Zenodo platform (https://zenodo.org/). Zenodo is an open data repository provided by CERN. In their own words: "Zenodo is an open repository for all scholarship, enabling researchers from all disciplines to share and preserve their research outputs, regardless of size or format. Free to upload and free to access, Zenodo makes scientific outputs of all kinds citable, shareable and discoverable for the long term." (https://www.openaire.eu/zenodo-guide).

We plan to create a NEANIAS community in Zenodo, so that all published datasets are grouped under this community. Zenodo supports basic metadata attributes (Dublin Core, see https://dublincore.org/) necessary to create a "record", i.e. an entry in the metadata

repository of Zenodo. It also supports some extra metadata attributes, specific to certain record types. For example, a journal publication record may contain the volume number. Each published upload should contain a DOI (persistent identifier), thus Zenodo assigns a DOI to each upload, if not provided by the user. Finally, Zenodo offers a REST search API for the user's own records (https://developers.zenodo.org/#rest-api), and also harvesting functionality through the OAI-PMH protocol (https://developers.zenodo.org/#oai-pmh).

The Zenodo service is already fully operational and at TRL9.

5.4. Data Validation Service

Short Name	DVS	**Lead partner**	NKUA
Type	Web service / Web app	**Contributors**	
Title	NEANIAS Data Validation Service		
Master element	N/A		
Description	The Data Validation Service (DVS) is a service used by the NEANIAS thematic services (and via them, by their end-users) to rate (grade) the datasets they use and obtain the ratings of datasets. The identifier of a dataset is its DOI. This service has an administrative interface that allows its administrators to define the dimensions for the rating and view all registered ratings. Additionally, this service is accessible by the thematic NEANIAS services via a REST API, allowing the calling services to create new dimensions for rating, to register ratings for datasets, and to obtain aggregate ratings for datasets. Finally, we will investigate the potential for automatic rating of datasets, wherever this is possible, after consulting with the corresponding users of each dataset format.		
Technical details	The service will provide a web UI for initialization, and for assisting in technical support. Its main usage is via its REST API. Consuming services will need to authenticate themselves first, and also pass along the authenticated end-user credentials that DVS will verify.		
Core integration	It needs the following integration • Authentication service, to identify the logged in user • Service instance registry, to register itself		
Depends on	-		
Use cases	a. A service creates a new "dimension" (subject) for rating datasets. b. A service retrieves all registered dimensions. c. A service registers a new rating for a dataset.		

	d. A service retrieves the aggregate rating of a dataset. b. Support personnel logs in the UI of the service to view all registered ratings and response to support requests.		
EOSC services integration	Eventually, EOSC AAI Potentially, EOSC portal, if service is deemed publishable		
EOSC Service	Potentially		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	NKUA	**License**	Apache License
First availability	2020-08 First version deployed with a local database, REST interface only, provided to support integration with other services.		
MS4 expectation	Prototype deployment, REST interface operational, without automated rating, integrated with NEANIAS AAI.		
MS6 expectation	Both Web UI and REST interfaces deployed and operational, without automated rating.		
MS7 expectation	If deemed publishable to EOSC, it will be fully integrated to EOSC at this point. Additionally, if automated rating proves viable, it will be operational at this point.		

5.5. Common User Interface Components

Short Name	NEANIAS web toolkit	**Lead partner**	CITE
Type	UI Component	**Contributors**	INAF
Title	NEANIAS web UI toolkit and template		
Master element	N/A		
Description	A template for all services UI to be reused by all partners exposing UIs in the context of NEANIAS (look-and-feel, logo, CSS, menus, etc.)		
Technical details	NEANIAS spans UIs from several independent service providers. Those may be implemented in different technologies however their web aspect, wherever present, will be utilizing standard technologies, such as HTML5/JS/CSS. NEANIAS web toolkit will offer optional elements and guidelines for all other services' web UIs to adopt in order to homogenize their look, and if possible, feel. The toolkit will include coloring guidelines, texts, html elements, proposals for menus, generic icons and, if needed, widgets for consistently exposing some backend services features.		

	The components will be based on existing mature web technologies, restyled and adapted to NEANIAS needs.		
Core integration	No general core integration is foreseen. Independent components may opt to communicate with specific backend services.		
Depends on	None		
Use cases	The toolkit elements will be usable by all users interacting with NEANIAS services via their web interfaces. However, their exploitation is targeted for service UI implementers. Examples of supported cases are: - Adopt the color guidelines for the NEANIAS service UI. - Include a common area for project information and NEANIAS gateway exploration - Adopt a specific set of icons for use by service UI. - Adopt a generic menu behavior and style. Etc		
EOSC services integration	None		
EOSC Service	NO		
Start TRL	TRL9	**Target TRL**	TRL9
IPR	CITE	**License**	FOSS (MIT, Apache or other similar)
First availability	2020-05 Generic stylesheet and color guidelines.		
MS4 expectation	Generic components for information presentation		
MS6 expectation	Specific components for service exposure. Additional UI elements, such as buttons and menus.		
MS7 expectation	Fine-tuned UI elements.		

5.6. NEANIAS Access Gate

Short Name	Catalogue Portal (C1.1)	**Lead partner**	ATHENA
Type	UI web app	**Contributors**	-
Title	NEANIAS Catalogue Portal		
Master element	Open-Science lifecycle support services implementation		
Description	NEANIAS catalogue portal will offer to users a single-entry point to all NEANIAS list of services, including core, thematic services as well as datasets and other resources. The catalogue portal will offer		

	functionality for users to *browse* the available services, *search and filter* along EOSC compliant classification, such scientific domain or category a service is most relevant to, the maturity of the service, the intended use, etc., as well as to compare NEANIAS service offerings.		
Technical details	Web portal built with AngularJS. Usage Analytics are collected in Matomo[6]. It integrates with EOSCoo AAI for authentication and it communicates with the catalogue service (c1.2) with rest api. https://github.com/eInfraCentral.		
Core integration	It needs the following integration • Authentication / Authorization for authenticated users to login and update service metadata. • Logging is needed for web analytics. Matomo FOSS is used for this reason. • It is already integrated (via REST API with Catalogue service C1.2)		
Depends on	Catalogue service C1.2 EOSC AAI		
Use cases	e. End users access the catalogue to find, access and browse services, datasets and other NEANIAS research resources c. NEANIAS service providers access the catalogue to add and update service metadata.		
EOSC services integration	EOSC AAI EOSC portal, www.eosc-portal.eu		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	NKUA \ ATHENA RC	**License**	Apache License
First availability	2020-08. Planned month of preliminary release to support NEANIAS outreach and engagement plan and brief reference of what may be usable at the point. No significant integration is expected.		
MS4 expectation	Prototype deployment. Not populated with NEANIAS service metadata. EOSC integration is not mandatory for EOSC services yet.		
MS6 expectation	Deployed and populated with all NEANIAS service metadata.		
MS7 expectation	Issue fixing and improvements.		

5.7. OpenDMP / Argos

Short Name	OpenDMP	**Lead partner**	CITE
Type	UI Web App / Web Service	**Contributors**	ATHENA, OpenAIRE
Title	OpenDMP Software		

[6] https://matomo.org/

Master element	N/A		
Description	OpenDMP is a software system that allows researchers to create and distribute interoperable, actionable Data Management Plans. The service is provided by OpenAIRE as Argos.		
Technical details			
Core integration	OpenDMP may integrate NEANIAS AAI. Yet supported deployment will be using EOSC AAI directly.		
Depends on	OpenDMP does not depend on other NEANIAS services.		
Use cases	NEANIAS researchers collaborate on the creation of Data Management Plans on OpenDMP.		
EOSC services integration	OpenDMP may integrate with - EOSC AAI - EOSC PID services (Zenodo etc) - EOSC repositories (Zenodo etc) - OpenAIRE indexing services		
EOSC Service	No		
Start TRL	TRL8	**Target TRL**	TRL9
IPR	OpenAIRE	**License**	FOSS (MIT, Apache or similar)
First availability	2020-02 Public release to accommodate observations of NEANIAS users. Deployed on OpenAIRE as Argos.		
MS4 expectation	Integration with EOSC AAI.		
MS6 expectation	Integration with PID providers.		
MS7 expectation	Final release.		

5.8. Data Publishing Service

The NEANIAS Data Publishing service is responsible for making research products available to the public. For the Data Publishing Service (DPS) NEANIAS will use the OpenAIRE's Zenodo platform (https://zenodo.org/).

When creating a Zenodo "record", a user may upload the research product (e.g., a dataset) to the platform. While the metadata of a record is open to allow harvesting via the OAI-PMH protocol, the data may be:

1. open under Creative Commons licensing,
2. embargoed under Creative Commons licensing,
3. restricted with per request access,

4. closed.

To support the case where a user aims to keep the data of the research product in a private repository, while still making its existence public via the Research Product Catalogue, we also propose the following convention. The user creates a Zenodo upload providing all necessary metadata and uploads a single JSON file with two data fields: a URL and a SHA256 hash of the file's contents. This way, metadata are searchable via the Zenodo platform, but the data remains private, possibly inside the NEANIAS Data Depositing Service. When a user wants to access this dataset, after being authorized via the Data Sharing Service, the consuming service can retrieve the URL from the metadata repository and verify the data is still intact, by comparing the hash of the data with the published one.

The Zenodo service is already fully operational and at TRL9.

5.9. Persistent Identifier Service / Zenodo

The NEANIAS Persistent Identifier Service will enable the generation of PIDs for digital assets. The goal of this service is to make NEANIAS digital assets citable and thus discoverable. The service will be integrated with all NEANIAS services. The NEANIAS PID Service will be provided by OpenAIRE's Zenodo open access repository (https://zenodo.org/).

Zenodo offers a list of REST methods to upload and publish research outputs assigning a Digital Object Identifier (DOI) to those outputs if they do not have one already. As already described in paragraphs 5.3 and 5.8, all NEANIAS research and data outputs will be published in Zenodo and thus be assigned a DOI that will allow them to be easily and unambiguously cited.

The Zenodo service is already fully operational and at TRL9.

6. C2 EOSC hub, RIs and cloud integration enabling services reference

C2 services, often called "Infrastructure enabling services ", or "enabling" services in the context of NEANIAS, form the lower level of services in the NEANIAS ecosystem and are the ones that deliver access to various levels of resources that serve the use cases of the project.

In this section C2 group of services is presented. Following the brief presentation of their, each service is presented in alignment with the Service Description template adopted by the project.

6.1. The role of C2 services in NEANIAS

C2 cluster of services encloses a set of loosely coupled elements that enable the assembly of a virtual infrastructure to serve other services, bringing closer to them resources and processes EOSC hub, other Research and Cloud Computing Infrastructures. C2 services, along with physical infrastructures and low-level resource abstractions, form the NEANIAS virtual infrastructure fabric, upon which all other services build.

C2 services make heavy use of existing services and software, launching off from a high TRL, and are re-engineered in order to adapt to the NEANIAS and EOSC information and service model. In few cases service software is already present yet adaptation of its data models and/or configuration is required in order to serve NEANIAS needs.

As a general observation C2 services are not expected to be exposed as end-user services in EOSC Marketplace, however quite a few offer valuable toolkits for other infrastructures to build upon, thus they will be openly shared with others, both in the form of services as well as free and open source software.

Note: C2 service range has substantially been extended compared to DoA presented ones, in order to respond to practical challenges raised by other services requirements and infrastructures' integration needs.

6.2. NEANIAS AAI

Short Name	NEANIAS AAI	**Lead partner**	CITE
Type	Web service / UI web app	**Contributors**	CITE
Title	NEANIAS Authentication & Authorization Infrastructure Service		
Master element	N/A		
Description	The NEANIAS AAI service will offer a horizontal solution for all NEANIAS services to cover the common requirements of authenticated access, whether these come in the form of direct user requests or as cross service invocations. Furthermore, provisions to support some degree of the authorization of these requests will be offered. The solution will be based on widely accepted protocols and standards to ensure wide applicability of the approach.		

Technical details	The NEANIAS AAI Service will be backed by the Open Source Keycloak (keycloak.org) application. The application will consist of a set of web API accessible endpoints to complete the needed authentication protocol interactions, as well as a set of web graphical user interfaces to support login, profile managements and administrative operations. To support the full lifetime of a service, three realms will be setup to cover authentication and authorization needs for: • Development • Staging • Production At each one of these realms, different external identity providers can be supported depending on needs. With respect to Authentication, the identity federation paradigm (https://openid.net/specs/openid-connect-federation-1_0.html#rfc.section.1) will be used to facilitate the authentication and registration to the NEANIAS catalog of users, principals that are authenticated through external identity providers. Comprising the identity federation, the following cases can be identified: • NEANIAS Single Sign On (SSO) • EOSC AAI • eduGAIN • OpenID Connect Identity Providers • Social platforms • Local users For further information one IdPs supported, one may refer to D6.1 §**Error! Reference source not found.** "**Error! Reference source not found.**" With respect to the supported grant flow through which the requestor identity is federated and subsequently propagated to servicing endpoints, the following grant flows will be supported: • Authorization Code Grant Flow (https://tools.ietf.org/html/rfc6749#page-24). • Client Credentials Grant • API Keys. For further information on grant flows, one may refer to D6.1 §**Error! Reference source not found.** "**Error! Reference source not found.**" The authentication flow that passes through the centrally managed NEANIAS AAI service will expose suitable user info endpoints (https://openid.net/specs/openid-connect-core-1_0.html#UserInfo) through which the respective id tokens can be used to retrieve the caller's claims. This information will be provided in the form of a JWT

	token (JSON Web Token) (https://tools.ietf.org/html/rfc7519) containing information to be utilized by servicing components. With respect to authorization, the following alternatives will be offered through the NEANIAS AAI Service. It should be noted that these alternatives are not necessarily mutually exclusive, and a combination can be used by some service as seems most fitting. • Role based • Resource based • Service Specific. For further information on authorization options, one may refer to D6.1 §**Error! Reference source not found.** "**Error! Reference source not found.**"		
Core integration	The NEANIAS AAI Service will integrate with a set of other core services to facilitate primarily monitoring and accounting of its usage. Integration with these services will be in a decoupled form, making this integration optional in setups that do not require such functionality: - Accounting – Accounting information on users registered, logins, user organization provenance, etc. - Logging – Logging information on low level troubleshooting as well as higher level action auditing		
Depends on	- External IdPs (optional)		
Use cases	- A NEANIAS service interact with the service to obtain user identification information - A NEANIAS service interacts with the service to obtain authorization assertions for a specific user		
EOSC services integration	EOSC AAI Service – The NEANIAS AAI Service will integrate with the respective EOSC AAI service and act as an Identity Consumer. Users registered and trusted by the EOSC AAI will be authenticated by the respective NEANIAS AAI service.		
EOSC Service	No		
Start TRL	TRL7	**Target TRL**	TRL9
IPR	CITE	**License**	Apache 2.0
First availability	2020-04: At this preliminary release, it is expected that at a minimum the dev and staging realms will be supported. Role based authorization will be available at the realms and at the client level. Federated users will be limited to NEANIAS AAI		

MS4 expectation	Production realm will be made available. Alpha integration with NEANIAS Logging and NEANIAS Accounting services will be available.
MS6 expectation	EOSC AAI integration will be available. Enhancements on the integration with the NEANIAS Accounting Service. Alpha availability of fine-grained resource level authorization will be evaluated.
MS7 expectation	Final version of selected fine-grained authorization capabilities, documentation, Accounting and Logging integration

6.3. Configuration Management Service

Short Name	NEANIAS Configuration Service	**Lead partner**	CITE
Type	Software library / Web service / UI web app	**Contributors**	CITE
Title	NEANIAS Configuration Management Service		
Master element	N/A		
Description	The NEANIAS Configuration Management Service will provide a key value storage for storing configurations that will be used by NEANIAS's services. Configurations will be created or updated using an HTTP API that will be available.		
Technical details	The NEANIAS Configuration Management Service will be backed by a distributed, highly available system. The storage will be replicated across multiple nodes of the service to guarantee data redundancy. NEANIAS Configuration Management Service will operate as a key / value store. Data will be stored using a unique key and will be accessed using the same key. Restricted access will be offered (ACLs) in order to restrict the actions (read/write) clients can perform. The main interface to NEANIAS Configuration Management Service will be a RESTful HTTP API. All requests will require authentication. The API will be able to perform updates to specific keys. Multi-key updates will be able using a transactional mechanism. NEANIAS Configuration Management Service will provide a User Interface for Administrator to easily review and monitor the usage of the service.		
Core integration	The NEANIAS Configuration Management Service will integrate with a set of other core services to facilitate primarily monitoring and accounting of its usage. Integration with these services will be in a		

	decoupled form, making this integration optional in setups that do not require such functionality: - Accounting – Accounting information on storage usage, etc. - Logging – Logging information on low level troubleshooting as well as higher level action auditing		
Depends on	- Accounting Components		
Use cases	- A NEANIAS service contacts the configuration management service in order to detect base configuration information, such as: the endpoint and database name to connect to and the accounts to use to access it. - A NEANIAS service contacts the configuration management service in order to detect the model of operation it applies		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL7	**Target TRL**	TRL8
IPR	CITE	**License**	Apache 2.0
First availability	2020-07 Initial service delivery dependable for services to store and retrieve public configuration		
MS4 expectation	Major service delivery, integrated with AAI.		
MS6 expectation	Major service update supporting secrets.		
MS7 expectation	Minor fixes and updates if required – no functional changes		

6.4. Service Instance Registry

Short Name	NEANIAS Service Registry	**Lead partner**	CITE
Type	Software library / Web service / UI web app	**Contributors**	CITE
Title	NEANIAS Service Instance Registry		
Master element	N/A		
Description	The NEANIAS Service Instance Registry will provide a list of all NEANIAS services among with information regarding their location and health status. NEANIAS's Services will be registered dynamically, creating list of all available services and their instances. Service Discovery HTTP API will be available to provide real-time list of services, their location, and their health.		

Technical details	The NEANIAS Service Instance Registry will be backed by a distributed, highly available system. There will be multiple master and client nodes. All communications between master and client nodes will be encrypted. Master nodes are where data is stored and replicated. Multiple master node instances will be used to avoid data loss in failure scenarios. Client nodes will be responsible for monitoring the health status of the services that run in the node, and the also the node itself. Usage of client nodes will be optional, as services will be able to register directly as master nodes. They will be used then fine-grained information is needed to be monitored in order to determine the health status of the node. The main interface to NEANIAS Service Instance Registry will be a RESTful HTTP API. Authenticated services will be able to register, while others will be able to discover instances via this API. The API will be able to perform basic CRUD (create, read, update and delete) operations on services, and health checks. Discover queries will be executed and results will be responded in JSON format. NEANIAS Service Instance Registry will provide an administrative User Interface to browse the registered services and provide health check information for each service.		
Core integration	The Service Instance Registry will integrate with a set of other core services to facilitate primarily monitoring and accounting of its usage. Integration with these services will be in a decoupled form, making this integration optional in setups that do not require such functionality: - Accounting – Accounting information on queries executed, etc. - Logging – Logging information on low level troubleshooting as well as higher level action auditing		
Depends on	-		
Use cases	A service		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	CITE	**License**	Apache 2.0
First availability	2020-07 Initial service delivery dependable for services to store and retrieve targeted service entries		

MS4 expectation	Major service delivery, integrated with AAI with preliminary service instance model.
MS6 expectation	Major service with final service model.
MS7 expectation	Minor fixes and updates if required – no functional changes

6.5. Log Aggregation Service

Short Name	NEANIAS Logging	**Lead partner**	CITE
Type	Software library / Web service / UI web app	**Contributors**	CITE
Title	NEANIAS Log Aggregation Service		
Master element	N/A		
Description	The NEANIAS Logging service will provide an aggregation functionality through which NEANIAS services can accumulate logs that are generated in a distributed fashion in a single centralized repository through well-defined endpoints and library utilities facilitating the transformation, enrichment and indexing of the log entries. The central repository can be searched and further aggregated to produce meaningful traces of user activity and facilitate troubleshooting and action auditing. Additionally, the service will present specific client technologies for service implementers to easily generate accounting information records.		
Technical details	The NEANIAS Logging Service will be backed by an ELK stack. The ELK stack consists of an ElasticSearch (https://www.elastic.co/elasticsearch/) index backend that stores data and makes them searchable, Logstash (https://www.elastic.co/logstash/) that facilitates the transformation of log entries extraction of additional metadata and homogenization of logged entries, Kibana (https://www.elastic.co/kibana) that offers visualization, aggregation and filtering graphical user interface over the indexed entries. To facilitate the aggregation of log, the Beats (https://www.elastic.co/beats/) framework for data shipping will be used. Specifically, beats modules for both file as well as http transfer will be provided with supportive configuration to lower the integration barrier for all NEANIAS Service Providers. Templates for log entry formats will be provided along with the respective Logstash transformation templates that will extract a		

	limited common set of information. This way, a homogenized set of information will be available for log browsing and aggregation operations. A Kibana dashboard will be provided that will offer browsing, filtering and visualization capabilities. Access to the Kibana graphical application will be subject to authorization enforced by relevant access management plugins and a level of integration with the NEANIAS AAI will facilitate some degree of global policy enforcement through the central AAI service.		
Core integration	The NEANIAS Logging Service is largely an independent, core service offering its functionality orthogonally from other services. Still, it is possible to integrate it with a limited number of other NEANIAS Core services to enhance the usability of the service within the NEANIAS ecosystem. Integration with these services will be in a decoupled form, making this integration optional in setups that do not require such functionality: - Authentication / Authorization – Authentication for user and Service access can take place through the NEANAIS AAI service. Based on access token role mappings, the respective operations and data access will be made available		
Depends on	Type-specific Storage services.		
Use cases	In the occurrence of an incident, an administrator, inspects the aggregated logs of the service to identify the issue occurring to a distributed, potentially not controlled by her, service component.		
EOSC services integration	When EOSC presents a service for accounting information gathering, the service will adopt the interface and information model in order to supply the required accounting information.		
EOSC Service	No (decision may be revised)		
Start TRL	TRL7	**Target TRL**	TRL8
IPR	CITE	**License**	Mixed (Apache 2.0, Elastic License)
First availability	2020-06: At this preliminary release, it is expected that basic configuration will be available to allow for direct aggregation of messages. No user interface to browse the logs will be available		
MS4 expectation	File and http beats configuration to support distributed log aggregation, basic transformations to extract key characteristics		

MS6 expectation	Graphical user interface to browse and analyze log entries through a Kibana interface with direct access to the underlying index
MS7 expectation	Integration with the NEANAIS AAI to facilitate federated user role based mapping to visualization and possibly administrative operations

6.6. Accounting Service

Short Name	NEANIAS Accounting	**Lead partner**	CITE
Type	Software library / Web service / UI web app	**Contributors**	CITE
Title	NEANIAS Accounting Service		
Master element	N/A		
Description	The NEANIAS Accounting service will provide an aggregation functionality through which NEANIAS services can centrally log accounting information as it is gradually accumulated by their usage from the respective authorized clients. Different accumulation policies and granularity of information can be supported depending on the respective services. The information model will consist of globally defined key point indicators and the aggregation and reports generated based on these will be available for further usage, as appropriate. Although it is possible to foresee use cases where the aggregated information can be used for run time decision making, throttling or altering the scope of user requests to consulting services, the primary focus of the service is to aggregate and report on the information.		
Technical details	The NEANIAS Accounting Service will be backed by a high performant NoSQL data store to aggregate and track the aggregated accounting information. The information model that will be supported will be progressively refined. In order to comply with emerging suggestions within the EOSC environment, the following aspects of accounted information will be investigated. This list is comprised of a set of Service / Resource Level Targets and Performance Indicators towards which, relevant information can be gathered or be assisted in calculate by the respective service providers: • Cost • Requests • Users • Usage		

	• Capacity • Coverage • Availability • Reliability • Serviceability/Durability • Other Performance Indicator Name/Value During the lifetime of the services, relevant indicators will be accumulated with different granularity and time frames. During this process, information will gradually be eligible to be folded to create aggregations that will facilitate in the long-term performance and maintainability of the Accounting Service data structures. To visualize and report on the aggregated information, dedicated reporting utilities will be made available through a graphical user web interface. Access and authorization on the reported content will be limited based on the NEANIAS AAI service. The Accounting Service will offer only aggregating and reporting functionality. The option to expose endpoints to cater for run time decision making logic will be evaluated, based on rule templates and threshold setting logic. Calculated fields can be made available based on specific accounting model semantics.
Core integration	The NEANIAS Accounting Service is largely an independent, core service offering its functionality orthogonally from other services. It mostly acts as an aggregator of information. Still, it is possible to integrate it with a limited number of other NEANIAS Core services to enhance the usability of the service within the NEANIAS ecosystem. Integration with these services will be in a decoupled form, making this integration optional in setups that do not require such functionality: - Authentication / Authorization – Authentication for user and Service access can take place through the NEANAIS AAI service. Based on access token role mappings, the respective operations and data access will be made available - Logging – Logging information on low level troubleshooting as well as higher level action auditing
Depends on	Type-specific Storage services
Use cases	A service provider wants to identify how much CPU time a particular user activity has occupied in the system. A service provider wants to know how many invocations per day were performed by a particular user. A service provider wants to know how much short-term storage is used by a user of a particular service.

EOSC services integration	If EOSC releases an Accounting Service API and data model, the service will comply with its specification and interoperate with it.		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	CITE	**License**	FOSS approved license Mixed (Apache 2.0, Elastic License) / CITE Permissive License / Other
First availability	2020-10 At this preliminary release, it is expected that basic ability to register accounting information according to NEANIAS accounting model will be provided. Visualization of the information will be at the level of record.		
MS4 expectation	Refinements on common accounting models. Aggregation utilities to ship data efficiently. Basic user interface without reporting functionality		
MS6 expectation	Extended accounting model and basic reporting functionality		
MS7 expectation	Full accounting model and reporting functionality available		

6.7. Notification Service

Short Name	NEANIAS Notification	**Lead partner**	CITE
Type	Web service / UI web app / UI Component	**Contributors**	CITE
Title	NEANIAS Notification Service		
Master element	N/A		
Description	The NEANIAS Notification service will act as a generic notification provider for services that want to incorporate some sort of notification mechanism in their workflows. It will allow generic configuration and parametrization of notification templates, give configuration options for selected notification channels to users and allow easy integration with other services that want to use its functionality		
Technical details	The NEANIAS Notification service will offer its functionality as a backend service, accepting notification events from other services. These events will be interpreted through a set of pre-configured		

	notification flows and will be delivered to its recipients based on the respective profile enabled.
	A descriptive list of the available features as well as corresponding technical implications, are listed below:
	• Notification Channels
	  ○ In-App Notification
	  ○ Mail
	  ○ SMS
	• Notification Policies
	  ○ Global policies matching notification flows to acceptable notifier channels
	  ○ User level policies based on ordered user preferences
	  ○ Fall through available options depending on whether respective profile information is available
	• Notification Tracking
	  ○ Depending on channel features
	  ○ Track delivery status of notification
	• Retries
	  ○ Configuration number of retries per notification flow
	  ○ Probabilistic retry intervals with progressive configurable delays
	  ○ Options to omit retries after long period of time
	• Notification Templates
	  ○ Different templates per notification channel
	  ○ Support multiple locale
	  ○ Configurable and easily upgradable
	• Interface
	  ○ Web API integration endpoints
	  ○ Asynchronous / Queue messages
	  ○ Administration Web App
	  ○ User Interface widget to display in-app notifications

Core integration	The NEANIAS Notification Service is largely an independent, core service offering its functionality orthogonally from other services. It mostly acts as an event consumer from other services that want to use its functionality. Still, it is possible to integrate it with a limited number of other NEANIAS Core services to enhance the usability of the service within the NEANIAS ecosystem. Integration with these services will be in a decoupled form, making this integration optional in setups that do not require such functionality: - Authentication / Authorization – Authentication for user and Service access can take place through the NEANAIS AAI service. Based on access token role mappings, the respective operations and data access will be made available

	- Logging – Logging information on low level troubleshooting as well as higher level action auditing - Accounting – Accounting information will be maintained and shipped to the Accounting Service to account for the usage of the service resources and operations		
Depends on	-		
Use cases	Upon the completion of a long running task, a client service may request notifications to be sent to users to retrieve their results.		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL7	**Target TRL**	TRL8
IPR	CITE	**License**	CITE Permissive License / Other
First availability	2020-07: Initial version offering single channel communication. Integration entry points through subset of available channels.		
MS4 expectation	Extended notification templates. User profiling endpoints and notification preferences. Notification context		
MS6 expectation	Additional notification channels including mail and in-app notifications. Web user interface widget for in-app notification browsing		
MS7 expectation	User profile integrations for automated profiling		

6.8. Data Depositing service

Short Name	GARR Cloud Platform Object Store	**Lead partner**	GARR
Type	Web service / storage service	**Contributors**	GARR
Title	NEANIAS Object Storage Service		
Master element	N/A		
Description	The GARR Cloud Platform Object Storage Service will provide a RESTful API to store and retrieve objects (files) over the network and to manage object containers.		

Technical details	The GARR Cloud Platform Object Storage Service is provided through the OpenStack platform and offers a RESTful interface to store, retrieve and manage objects and containers (i.e. file and directory abstractions) and their metadata (a key=value format is supported). The GARR Cloud object storage service is based on Ceph, which provides the Rados Gateway daemon and implements transparent data redundancy and hardware failover. The GARR Cloud infrastructures are connected to the facilities of the international research community through the high-performance network links of the GÉANT network.		
Core integration	The GARR Cloud Platform Object Storage Service is an independent core service offering its functionality orthogonally from other services. It mostly acts as a consumer from other services that want to use its functionality.		
Depends on	-		
Use cases	- A user deposits or retrieves data contained in one or more files in consortium storage - A service deposits or retrieves data contained in one or more files in consortium storage		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL8	**Target TRL**	-
IPR	Ceph authors	**License**	LGPL, GPL, BSD
First availability	2019-11 The service is already available at the GARR Cloud Platform premises https://cloud.garr.it		
MS4 expectation	Service is present. If required configuration/allocation changes may occur.		
MS6 expectation	Service is present. If required configuration/allocation changes may occur.		
MS7 expectation	Service is present. If required configuration/allocation changes may occur.		

6.9. Data Sharing service

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Short Name	DaShaS	**Lead partner**	NKUA
Type	Web service	**Contributors**	
Title	NEANIAS Data Sharing Service		
Master element	N/A		
Description	The Data Sharing Service (DaShaS) is a service used by the NEANIAS thematic services (and via them, by their end-users) to allow data stored internally in the NEANIAS Data Storage Service (DSS) to be shared between the end users. It is an authorization service for data objects, that is expected to be consulted before servicing/using objects from the DSS.		
Technical details	The service will provide a REST API via which, consuming services will register grant/revocation of access rights on objects in the DSS. Consuming services will need to authenticate themselves first, and also pass along the authenticated end-user credentials that DVS will verify.		
Core integration	It needs the following integration • Authentication service, to identify the logged in user • Service instance registry, to register itself • In needs to be injected in the pipeline of obtaining an object from the NEANIAS Data Depositing Service, as an authorization step.		
Depends on	-		
Use cases	a) A service registers, on behalf of user A, the authorization for user B to use/read/write data object O stored in the NEANIAS DSS. b) A service registers, on behalf of user A, the revocation of a previously granted authorization for user B to use/read/write data object O. c) A service queries the access rights of user B for object O.		
EOSC services integration	Eventually, EOSC AAI Potentially, EOSC portal, if service is deemed publishable		
EOSC Service	Potentially		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	NKUA	**License**	Apache License
First availability	2020-08 First version deployed with a local database, provided to support integration with other services.		
MS4 expectation	Prototype deployment, integrated with NEANIAS AAI.		
MS6 expectation	Deployed and operational.		

MS7 expectation	If deemed publishable to EOSC, it will be fully integrated to EOSC at this point.

6.10. Data Transfer service

Short Name	Data Avenue	**Lead partner**	SZTAKI
Type	GUI / Web Service/ REST-API	**Contributors**	ALTEC
Title	Data Avenue		
Master element	N/A		
Description	Data Avenue is a data storage access tool developed by SZTAKI. Data Avenue aims at simplifying handling data residing on remote storages of various types, such as FTP servers, Amazon S3, Hadoop Distributed File System (HDFS), OpenStack's Swift, etc. Data Avenue is not a storage, it does not persist any data at all (neither access credentials), but it allows to upload, download, organize data (create folders/containers/buckets, delete files or entire directories, etc.), and even transfer data between any of the supported storage types, e.g. from S3 to HDFS.		
Technical details	Data Avenue offers a uniform and easy-to-use interface to access all the supported storage types. E.g., directory creation can be done in the same way regardless it happens on a GridFTP server or on Amazon S3 - from the user's point of view. There is lightweight, pure javascript-based graphical user interface (GUI) that can be used in any web browser, and there is a clean REST application programming interface (API).		
Core integration	This service needs integration with the following components: - Authentication / Authorisation - Accounting - Logging - Notification - Storage access (temporary / permanent)		
Depends on	- Storage Access - AAI		
Use cases	- User requests that data, residing on a repository (storage) covered by Data Transfer Service protocols, are transferred to another repository of compliant technology. - User requests that data from a personal repository (storage) are uploaded in the infrastructure for processing by a service.		

	- A service requests that data residing on the infrastructure are fetched next to a service for local processing in case the service has its compliant storage. - A service requests that data are uploaded to a storage service from the local storage of a service.		
EOSC services integration	- (research data repositories) - (AAI)		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	SZTAKI	**License**	Apache 2.0
First availability	2020-08 Deployed on NEANIAS infrastructure. The software is already available at https://github.com/SZTAKI-LPDS/data-avenue		
MS4 expectation	Initial deployment on GARR cloud and integration to Swift object store on GARR cloud happens.		
MS6 expectation	Integration to some of the core services such as AAI, Logging is available in the service.		
MS7 expectation	Final deployment; Integration to further core services such as Accounting, Notification and to all other necessary services. EOSC integration happens.		

6.11. Data exploration service

Short Name	Data Exploration	**Lead partner**	MEEO
Type	GUI / Web Service / API	**Contributors**	JACOBSUNI
Title	Data Exploration service		
Master element	N/A		
Description	The service allows the researchers to discover and explore a large variety and volume of geospatial datasets including Earth Observation and Planetary Science products driven by community needs. The service offers the following user interfaces: - the **Explorer**, a web-based graphic user interface to allow users to explore, access, process and download data. Explorer integrates also data processing functionalities and a dashboard for administrator to setup the application - The **adamapi** Application Processing Interfaces (APIs), that provide a python-library to directly access the ADAM data		

	access and processing capabilities directly integrated in the user's code and applications. - the **Jupyter Hub**, a web-based processing environment to allow users to import, write and execute code that runs close to the data, exploiting the the adamapi functionalities on a remote computation environment The core services allow at performing essential operations on selected data thanks to the advance user interfaces like Jupyter notebook and REST/API. The standard data access will allow the implementation of operational services, the notebook will allow to make basic and advanced processing on-line.		
Technical details	OGC services and OpenAPI backends will be accessed via Desktop and Jupyter/Python interfaces.		
Core integration	This service needs integration with the following components: - NEANIAS AAI Authentication / Authorization - Accounting		
Depends on	- Spatial data stores (C4.4) - External access services		
Use cases	- Users search for datasets, data products based on metadata, geographic location (S1, Ax) - Users request data subsets and obtain (S2, S3, Ax)		
EOSC services integration			
EOSC Service	No		
Start TRL	TRL8	**Target TRL**	TRL8
IPR	MEEO	**License**	Proprietary Software
First availability	2020-08 Initial deployment on NEANIAS infrastructure, with a set of data query able (e.g. Copernicus datasets)		
MS4 expectation	• Service setup, including integration with NEANIAS AAI component • Registration in the central service repository		
MS6 expectation	Showcase of data exploration. Logging available		
MS7 expectation	Final deployment		

6.12. Computational resources access service

Computational resource access service provides a gateway to workflow, distributed processing engines or other computational capacity abstractions that will allow computational demands of NEANIAS to be spanned to other RIs, EOSC hub services or consortium resources (Includes access to Apache Spark / Hadoop).

The functionalities of this service can be implemented based on existing cloud/container orchestration tools. The service will be responsible to instruct the orchestration engine for instantiating the necessary computational resource(s) on the target cloud and for building the required infrastructure/ reference architecture on demand. Once the reference architecture has been built, access to the reference architecture is provided for the user.

Short Name	GARR Cloud Platform Service	**Lead partner**	GARR
Type	Computational service, IaaS	**Contributors**	GARR
Title	GARR Cloud Platform Service		
Master element	N/A		
Description	The GARR Cloud Platform Service will provide NEANIAS services with computational facilities in the form of virtual machines.		
Technical details	The GARR Cloud Platform is based on OpenStack, an open source software suite aimed at implementing cloud services in datacenters. The GARR Cloud Platform is a multi-tenant environment spawning three different geographical regions. It is federated through the IDEM/eduGAIN AAI. The persistency layer relies on volumes backed by the Ceph system, which provides enhanced performance and data redundancy. The GARR Cloud infrastructures are connected to the facilities of the international research community through the high-performance network links of the GÉANT network.		
Core integration	The GARR Cloud Platform Service is an independent core service offering its functionality orthogonally from other services. It mostly acts as a consumer from other services that want to use its functionality.		
Depends on	-		
Use cases	- A user requests virtual machines for deployment of service. - Increased load of services leads to increase of allocated computational power.		

	- A service requests computational power to execute computations.		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL8	**Target TRL**	-
IPR	OpenStack Foundation	**License**	Apache License 2.0
First availability	The service is already available at the GARR Cloud premises: https://cloud.garr.it		
MS4 expectation	-		
MS6 expectation	-		
MS7 expectation	-		

Short Name	GARR Container Platform Service	**Lead partner**	GARR
Type	Computational service, PaaS	**Contributors**	GARR
Title	GARR Container Platform Service		
Master element	N/A		
Description	The GARR Container Platform Service will provide NEANIAS services with the orchestration of computational facilities (CPU and GPU) besides storage and networking resource abstractions.		
Technical details	The GARR Container Platform is based on Kubernetes, a popular open source platform for container orchestration which automates the deployment, management and scaling of applications. The GARR Container Platform is a multi-tenant environment whose authentication is managed by the GARR Cloud Platform OpenStack identity service (Keystone). The GARR Container Platform is deployed on bare metal to provide GPU resources which can be leveraged by Docker containers. The persistency layer relies on volumes backed by the Ceph system, which provides enhanced performance and data redundancy. The GARR Cloud infrastructures are connected to the facilities of the international research community through the high-performance network links of the GÉANT network.		

Core integration	Although the GARR Container Platform Service depends on the GARR Cloud Service for user authentication, from the NEANIAS point of view it is an independent core service offering its functionality orthogonally from other services. It mostly acts as a consumer from other services that want to use its functionality.		
Depends on	GARR Cloud Platform Service		
Use cases	- A user deploys a container-based service - A service spawns containers to execute specific tasks		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL8	**Target TRL**	-
IPR	Cloud Native Computing Foundation	**License**	Apache License 2.0
First availability	The service is already available at the GARR Cloud premises: https://cloud.garr.it/containers/		
MS4 expectation	-		
MS6 expectation	-		
MS7 expectation	-		

Short Name	GARR DaaS	**Lead partner**	GARR
Type	Computational service, PaaS	**Contributors**	GARR
Title	GARR Deployment as a Service		
Master element	N/A		
Description	The GARR DaaS will provide NEANIAS services with computational facilities and application abstractions.		
Technical details	The GARR DaaS is based on Juju, an open source platform which can be used to deploy, scale and manage applications implemented through the composition of microservices. GARR DaaS allows users to deploy a wide range of applications on top of OpenStack through a web interface. These applications include tools which are aimed at data processing, such as Apache Spark, Hadoop, and Elasticsearch.		

	The GARR Cloud infrastructures are connected to the facilities of the international research community through the high-performance network links of the GÉANT network.		
Core integration	Although the GARR DaaS depends on the resources provided by the GARR Cloud Platform Service, from the NEANIAS point of view is an independent core service offering its functionality orthogonally from other services. It mostly acts as a consumer from other services that want to use its functionality.		
Depends on	GARR Cloud Platform Service		
Use cases	- A user deploys an application - A service deploys an application to execute a specific task		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL7	**Target TRL**	-
IPR	Canonical Ltd.	**License**	GNU Adata
First availability	The service is already available at the GARR Cloud premises: https://cloud.garr.it/apps/daas/		
MS4 expectation	-		
MS6 expectation	-		
MS7 expectation	-		

Short Name	MiCADO	**Lead partner**	SZTAKI
Type	Computational service, PaaS	**Contributors**	SZTAKI
Title	MiCADO DaaS (Deployment as a Service)		
Master element	N/A		
Description	The MiCADO DaaS will provide NEANIAS services with computational facilities and application abstractions.		
Technical details	The MiCADO DaaS is based on Terraform, Occopus, Kubernetes and Prometheus, an open source platform which can be used to deploy, auto-scale and manage applications implemented through the composition of containers or virtual machines. MiCADO DaaS allows users to deploy a wide range of applications on top of various Cloud providers (AWS, GCE, Azure, Openstack, EGI		

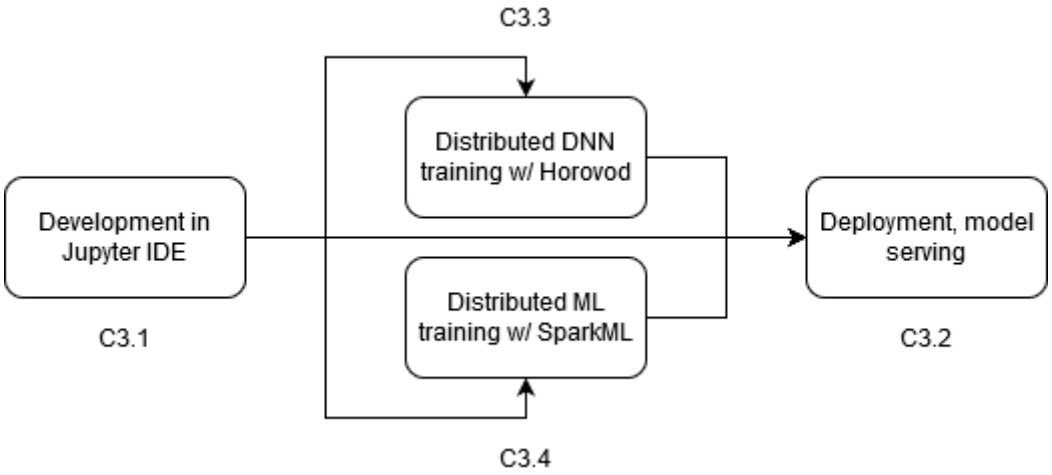
	Cloud) through an API. The API can be utilized through Jupyter notebook as well.		
Core integration	MiCADO DaaS Service is offering its functionality to allocate resources and deploy complex applications on them. This service is therefore independent from other services.		
Depends on	- AAI - Accounting - Notification - Logging		
Use cases	- (Micro)Service deployment hosted by dynamically instantiated virtual machines on GARR or other clouds. - Set of interacting (micro)services deployed on GARR or other clouds - Automatic scaling of (micro)service(s) and virtual machines running on GARR or other clouds including EGI		
EOSC services integration	-		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	SZTAKI	**License**	Apache 2.0
First availability	2020-08 Deployed on NEANIAS infrastructure. The service is already available at https://micado-scale.eu/.		
MS4 expectation	Initial deployment; MiCADO is deployed, several examples are developed in order to demonstrate how MiCADO can be used to automatically deploy and scale reference architecture		
MS6 expectation	Integrated showcasing; Integration to some of the core services such as AAI, Logging is available in the service		
MS7 expectation	Final deployment; Integration to further core services such as Accounting, Notification and to all other necessary services is available in MiCADO. EOSC integration happens.		

7. C3 Artificial Intelligence services reference

C3 Artificial Intelligence services form the upper level of core services in the NEANIAS ecosystem and provide facilities that may reach up to the end user offering features of a typical Machine Learning (ML) workflow lifecycle, and that are also intended for composition of higher level services.

In this section overall role of C3 services in NEANIAS is presented, then each service is described in alignment with the Service Description template adopted by the project.

7.1. The role of C3 services in NEANIAS



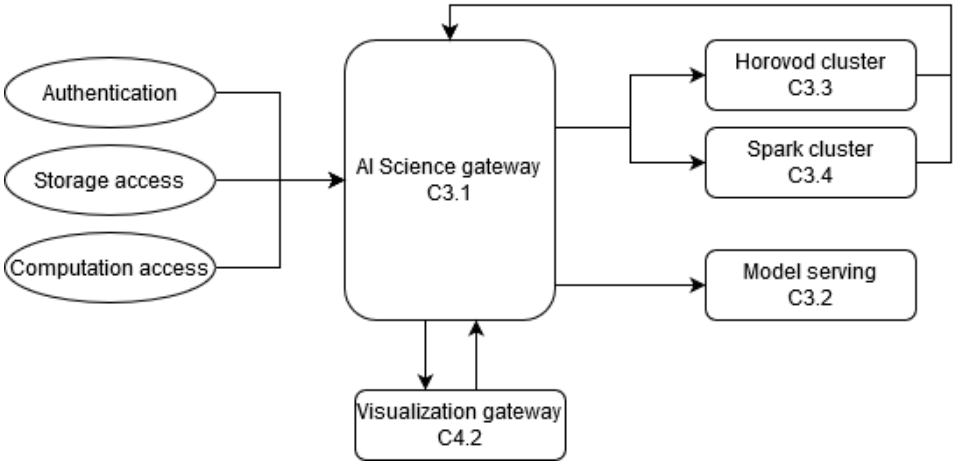
Each C3 service represents the elements of a typical ML workflow lifecycle: initially, the model needs to be designed and implemented. In case the load of data exceeds the capabilities of a single workstation, the training should be offloaded to a distributed computational cluster. Depending on the model, different approaches for parallelization exist: parallel processing of Deep Neural Networks is fundamentally different than the training of other machine learning models, which could be efficiently decomposed to independent pieces. After a model is trained, serving and deployment requires dedicated resources.

7.2. C3.1 AI Science Gateway: service for development of ML models using Jupyter Hub

A development environment is necessary for the researchers to design, prototype, implement and test AI powered solutions. To serve as a universal core service for multiple users, the popular IPython based Jupyter Hub project is preferred, with TensorFlow and Keras libraries.

Researchers are able to prototype solutions, while computationally intense training can be loaded to a computational cluster. Results can be visualized using the Visualization gateway (C4.2).

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

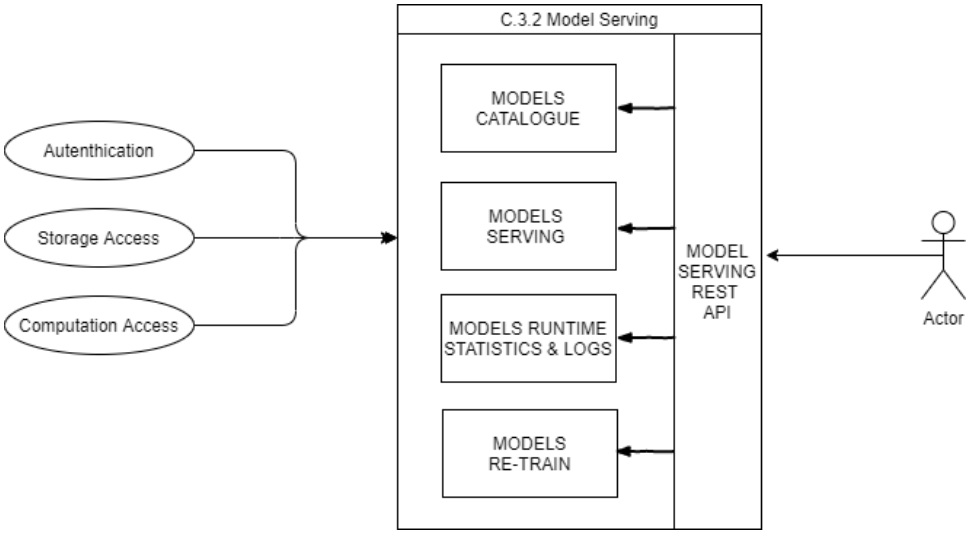


Short Name	Jupyter IDE (C3.1)	**Lead partner**	SZTAKI
Type	Web service / UI web app	**Contributors**	INAF
Title	AI Science Gateway: service for Development of Machine Learning Model using Jupyter Hub		
Master element			
Description	A web-based development environment for scientists to design, develop and evaluate machine learning solutions. After development, training of large models on big data can be done in a distributed environment, refer to C3.3 and C3.4.		
Technical details	The core technology is Python, which is the most popular programming language in machine learning and data science scenes. Jupyter Hub is a multi-user development environment using the IPython interface, allowing code implementation and interpretation using a web interface. The kernel used will include TensorFlow and Keras (www.keras.io), as the most popular libraries for neural network development, and other relevant packages as well. While the core is available as open source, some development on integrating the authentication protocols, and integrating relevant functions of other services of the machine learning lifecycle (design, train, deploy).		
Core integration	Integration with the following is necessary: - Authentication / Authorization - Storage access (temporary / permanent) - Computation access (direct or indirect) - Visualization		

Depends on	C2 6.12 Data exploration service C2 6.13 Computational resources access service		
Use cases	• Users of thematic services to access and develop machine learning models • Prototyping of machine learning algorithms		
EOSC services integration	To be investigated later		
EOSC Service	No (decision may be revised)		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	SZTAKI	**License**	FOSS, such as Apache
First availability	2020-06, initial version with core Python packages		
MS4 expectation	Prototype deployment for showcasing; basic IDE functionality		
MS6 expectation	Integrated showcasing; integration of some core services		
MS7 expectation	Final deployment		

7.3. C3.2 Serving trained ML models

Accessing trained models as web services should be possible seamlessly, with low efforts. To deploy a model as an API, large amount of memory is necessarily allocated and used constantly. Technically, there are multiple approaches to implement model serving, user and researcher-friendliness is prioritized.



The fundamental idea behind model serving is that authorized clients are able to access prediction functionality without directly accessing the model, which would create a serious

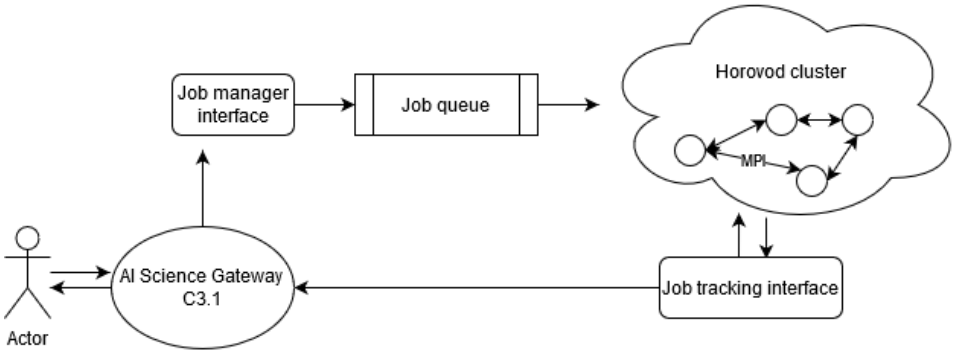
overhead caused by data transfer and memory allocation. Instead, a RESTful API based communication is preferred between a model server and the client. The server implements the model, allocates memory to efficiently handle prediction, and listens for input parameters on the interface. On request, prediction is done using the inner implementation, and the results are responded to the client.

Short Name	Model serving (C3.2)	**Lead partner**	ALTEC
Type	Web service	**Contributors**	SZTAKI, UNIMIB, INAF
Title	Serving trained ML models		
Master element			
Description	Serving trained machine learning models as a web service.		
Technical details	A web interface with a REST API, which receives the input parameters and replies with the prediction(s). Implementation using TensorFlow Serve or Cortex currently seems suitable. On designing, the service must fit to C3.1 (and possibly C3.3 & C3.4), as the users should be able to deploy the designed and trained models seamlessly.		
Core integration	This service will probably need integration with: - Authentication / Authorisation - Storage Access - Computation Access		
Depends on	C3.1 Jupyter IDE Might depend on C3.3, C3.4 C2 6.12 Data exploration service C2 6.13 Computational resources access service		
Use cases	• Serving of trained models for production		
EOSC services integration	To be investigated later		
EOSC Service	No (decision may be revised)		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	ALTEC	**License**	FOSS, such as Apache
First availability	2020-09, initial version with core model serving capabilities based on open source solution.		
MS4 expectation	Prototype deployment for showcasing; experiment level.		

MS6 expectation	Integrated showcasing; integration of some of the core services.
MS7 expectation	Final deployment

7.4. C3.3 Distributed Multi-GPU training of large ML models using Horovod

The training of deep neural networks on large amount of data requires a significant amount of processing time, which could be reduced by using GPU accelerators. The next step in increasing the performance is by using multiple GPU-enabled nodes in a distributed environment. While the parallel processing of deep neural networks has its limitations, Horovod (https://eng.uber.com/horovod/) is a robust tool which can be used to scale the training of TensorFlow based network implementations.



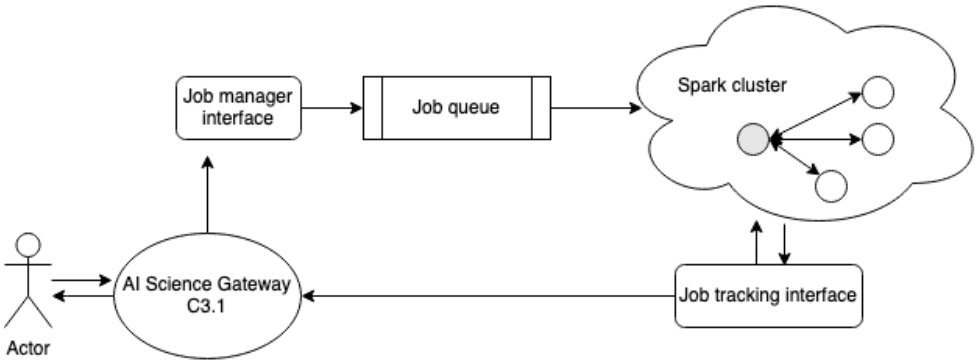
Short Name	Horovod cluster (C3.3)	**Lead partner**	SZTAKI
Type	Web service	**Contributors**	INAF, UNIMIB
Title	Distributed Multi-GPU training of large ML models using Horovod		
Master element			
Description	The training of deep neural networks on big data takes significant time, even on GPU-enabled workstations. To increase efficiency, a distributed computation cluster should be used, where users can define models to train and collect results.		
Technical details	A Horovod based architecture of GPU-accelerated nodes, where the communication is based on the MPI protocol. The training jobs are queued, resulting model parameters and training logs are served to the user. Real-time tracking of the training process is possible, e.g. using TensorBoard.		

Core integration	- Data discovery / Data publication - Storage access (temporary / permanent) - Computation access (direct or indirect)		
Depends on	Might depend on C3.1		
Use cases	Scaled processing of computationally intense Deep Neural Network training		
EOSC services integration	To be investigated later		
EOSC Service	No (decision may be revised)		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	SZTAKI	**License**	FOSS, such as Apache
First availability	2020-08, initial version		
MS4 expectation	Prototype deployment for showcasing; experiment level		
MS6 expectation	Integrated showcasing; integration of some of the core services		
MS7 expectation	Final deployment		

7.5. C3.4 Distributed Machine Learning using SparkML

Scaling up machine learning solutions other than deep neural nets (both for sake of classification, as well as for other tasks like regression and clustering) can be carried out using Apache Spark framework. The communication method and the data processing toolkit is a mature solution in distributed processing, and the ML library is built on these fundamental blocks. The Spark ML lib has API to Python, Java and R.

The structure of this service will be analogous as service C3.3, in an attempt to provide a uniform approach, programming interface and model to the developers employing these services.



Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Short Name	Spark cluster (C3.4)	**Lead partner**	UNIMIB
Type	Web service	**Contributors**	SZTAKI
Title	Distributed Machine Learning using SparkML		
Master element			
Description	In addition to models based on neural networks (that can benefit from a multi-GPU deployment, especially for learning phases) that are managed through C3.3, other AI techniques for classification, regression, or clustering tasks can however benefit from a deployment and execution in a distributed and memory-based architecture such as the one offered by Spark. In particular, C3.4 will support a variety of machine learning models such as Support Vector Machines, Decision Trees and ensemble models (Random Forests and Gradient Boosting) for both classification and regression, clustering algorithms, and other support functions that are useful in real-world, large scale, machine learning pipelines. The set of basic ML algorithms and models can also be extended by means of creative parallel computing approaches (e.g. parallel implementations of DBSCAN algorithm on top of Spark, an algorithm which is not natively provided by MLib, are known and available).		
Technical details	Apache Spark cluster with MLlib, using a job queue, and transferring results and logs back to the users similarly as in C3.3.		
Core integration	- Data discovery / Data publication / Data Transfer (/ Data Exploration?) - Storage access (temporary / permanent) - Computation access (direct or indirect)		
Depends on	Might depend on C3.1		
Use cases	Scaled training of machine learning models, where the Horovod cluster is not applicable		
EOSC services integration	To be investigated later		
EOSC Service	No (decision may be revised)		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	SZTAKI	**License**	FOSS, such as Apache
First availability	2020-08, initial version		
MS4 expectation	Prototype deployment for showcasing; experiment level		

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

MS6 expectation	Integrated showcasing; integration of some of the core services
MS7 expectation	Final deployment

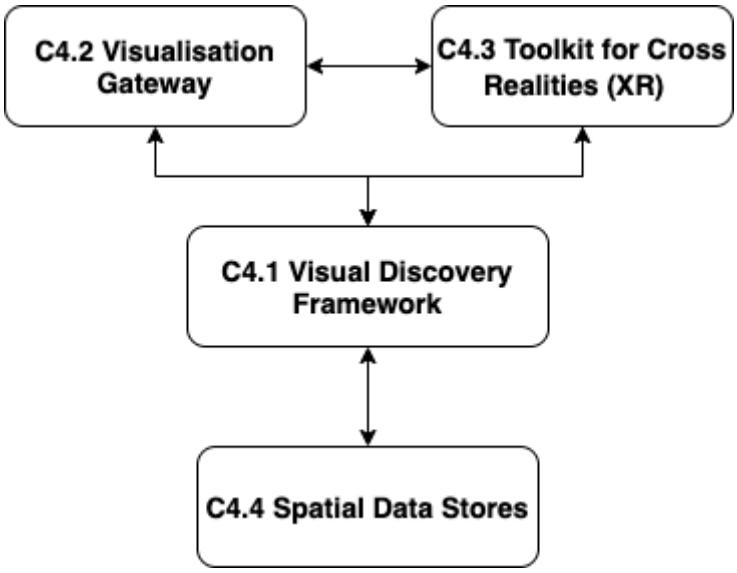
8. C4 Visualisation services reference

C4 services, namely "visualisation services", in the context of NEANIAS, form the upper level of core services in the NEANIAS ecosystem and provide facilities that may reach up to the end user offering features of a typical visualisation workflow lifecycle, and that are also intended for composition of higher level services.

In this section overall role of C4 services in NEANIAS is presented, then each service is described in alignment with the Service Description template adopted by the project.

8.1. The role of C4 services in NEANIAS

Each C4 service represents the elements of a typical visualisation workflow lifecycle with focus on visual discovery frameworks (C4.1, VD-section 8.2), science gateways (C4.2, VG-section 8.2), a cross realities framework to support VR/AR (C4.3, XR-section 8.3), and spatial data stores (C4.4, DS-section 8.4). The spatial data stores service underpins the visual discovery framework whose functionality is exposed via workflows through the science gateway or via advanced interface mechanisms through the XR toolkit; the overall schema is as below.



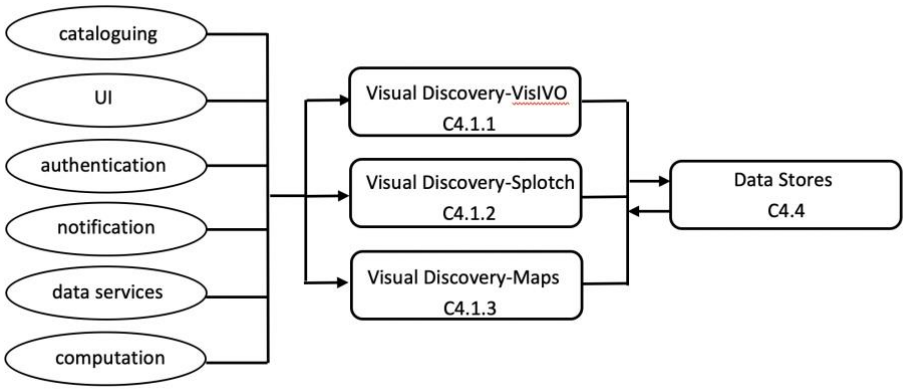
C4 services schema

C4 activities will identify underlying common themes and synergies, and apply cross-cutting glue and abstraction mechanisms, to realize a solid, modular, cross-sector, open-source, core pool of methods and tools to support visualisation services, leveraging on the resources available from RIs and EOSC-hub facilities. C4 will establish, implement and test such services to underpin a range of multi-faceted visualisation workflows, spanning from 2D/3D spatio-temporal to composite 2D/3D visuals of high dimensionality for computationally demanding cross reality applications, e.g. supporting mechanisms for scalable visualisation or production

of assets for exploitation within game engines through VR headsets and advanced motion tracking systems. When the load of data exceeds the capabilities of a single workstation, the workflow computation will be offloaded to distributed computational clusters. Depending on the tools deployed, different approaches for parallelization will be exploited, in order to harvest optimally different types of underlying infrastructures, ranging from small-scale clusters with or without GPUs to large-scale heterogeneous architectures. Although the core functionalities developed in C4 services will allow customisation to serve specific needs of NEANIAS end-user communities, they are also expected to be able to generalise sufficiently over the core elements of thematic services to underpin demands of other EOSC end-user communities to reach far beyond NEANIAS. At start, all services will be founded on existing TRL6 software solutions, which are envisaged to be refined progressively to reach at least TRL 8 in the course of NEANIAS.

8.2. C4.1 Framework for Visual Discovery (VD)

C4.1 will provide tools to support data intensive computing for visual scientific discovery (including research training and outreach) for observational data, theoretical simulations and 2D/3D tiling and maps. C4.1 will consist of three distinct high-performance services for visual analytics from multidimensional data tables (VD-VisIVO), high-quality volume rendering of particle-based datasets exploiting a variety of parallel programming models leveraging upon heterogeneous infrastructures (VD-Splotch) and creation of interactive 2D/3D tilings and maps (VD-Maps). Details on these services are summarised in the tables below.



C4.1 services diagram

Short Name	VD-VisIVO (C4.1.1)	**Lead partner**	INAF
Type	Software library / Web Service / Command Line Interface	**Contributors**	UoP
Title	Visual Discovery Framework - VisIVO		

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

Master element	N/A		
Description	The service offers a framework for data intensive visual discovery for experiments and data analysis (including support for training of researchers and public outreach) tailored to the requirements of NEANIAS while reaching out to other EOSC end-user communities.		
Technical details	The service will make available for data processing and visual discovery the suite of tools provided by VisIVO. VisIVO is specifically designed for distributed computing environments such as cloud infrastructures and offers a framework for exploration of large-scale scientific data-sets creating customized views of 3D renderings from multi-dimensional data tables from various sources.		
Core integration	It may need the following integration: • Catalogue service (monitoring) • Common user interface components • AAI Authentication & Authorization Infrastructure Service • Notification service • Data services (depositing, sharing, transfer, publishing, exploration) • Computational resources access service		
Depends on	It may depend on: • C4.1.2 VD-Splotch • C4.4 DS		
Use cases	• Within Neanias, all space thematic services (S1, S2, S3). • General users with multi-dimensional data in tabular formats (e.g. ASCII, CSV, VOTable, FITS, and others) wanting to explore data into multi-dimensional visual renderings.		
EOSC services integration	The service may be integrated with EOSC Cloud Container Computing services.		
EOSC Service	TBC		
Start TRL	TRL6	**Target TRL**	At a minimum TRL8
IPR	INAF/UoP	**License**	FOSS such as GNU General Public License (GPL).
First availability	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan. Release of the software as Docker containers.		

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

MS4 expectation	First version for service showcasing deployed within NEANIAS infrastructure (M14).
MS6 expectation	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.
MS7 expectation	Final deployment (M32).

Short Name	VD-Splotch (C4.1.1.2)	**Lead partner**	UoP
Type	Software library / Web Service / Command Line Interface	**Contributors**	INAF, SZTAKI
Title	Visual Discovery Framework - Splotch		
Master element	N/A		
Description	The service offers a framework for data intensive visual discovery for experiments and data analysis (including support for training of researchers and public outreach) tailored to the requirements of NEANIAS while reaching out to other EOSC end-user communities.		
Technical details	The service will make available for data processing and visual discovery the suite of tools provided by Splotch. Splotch supports very large-scale datasets and an array of diverse parallelisation models for fast, high-quality distributed particle volume rendering of particles coming from simulations, e.g. smoothed-particle hydrodynamics simulations and other sources.		
Core integration	It may need the following integration: • Catalogue service (monitoring) • Common user interface components • AAI Authentication & Authorization Infrastructure Service • Notification service • Data services (depositing, sharing, transfer, publishing, exploration) • Computational resources access service		
Depends on	It may depend on: • C4.4 DS		
Use cases	• Within NEANIAS, all space thematic services (S1, S2, S3).		

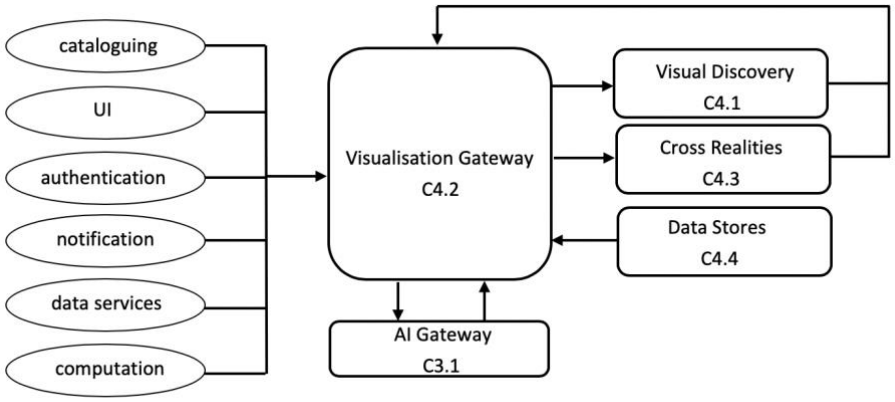
	• General users with cosmological simulation formats (e.g. Binary, Gadget, HDF5, Tipsy, and others) wanting to explore data into high-quality particle-based visual renderings.		
EOSC services integration	The service may be integrated with EOSC Cloud Container Computing services.		
EOSC Service	TBC		
Start TRL	TRL6	**Target TRL**	At a minimum TRL8
IPR	UoP	**License**	FOSS such as GNU General Public License (GPL).
First availability	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan. Release of the software as Docker containers.		
MS4 expectation	First version for service showcasing deployed within NEANIAS infrastructure (M14).		
MS6 expectation	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.		
MS7 expectation	Final deployment (M32).		

Short Name	VD-Maps (C4.1.1.3)	**Lead partner**	CORONIS
Type	Web Service / Command Line Interface	**Contributors**	JACOBS
Title	Visual Discovery Framework - 2D/3D Tiling and Maps		
Master element	N/A		
Description	The service consists of a set of tools allowing the visualisation of large-scale large-resolution maps, either 2D (images), 2.5D (elevation/bathymetric maps) or 3D (3D meshes), in real-time on a web app. First, a conversion service transforming common single-file formats into multi-resolution tiled data structures will be provided. Then, we will also provide a generic web viewer using the created data structures.		
Technical details	The conversion service translating single file formats into a hierarchical data structure will be provided in the form of a set of		

	command line tools within a container, and the resulting data structures will comply with OGC standards. Then, the web app allowing visualisation of such data structures will be implemented using CesiumJS library (www.Cesium.com/cesiumjs/).		
Core integration	It may need the following integration: • Catalogue service (monitoring) • Common user interface components • AAI Authentication & Authorization Infrastructure Service • Notification service • Data services (depositing, sharing, transfer, publishing, exploration) • Computational resources access service		
Depends on	It may depend on: - C4.1.1 VD-VisIVO - C4.1.2 VD-Splotch - C4.4 DS		
Use cases	- Within NEANIAS, all underwater thematic services (U1, U2, U3). - General users with GIS data in common formats (GDAL-compliant) wanting to convert their data into multiresolution tiled data structures.		
EOSC services integration	The service may be integrated with EOSC Cloud Container Computing services.		
EOSC Service	TBC		
Start TRL	TRL6	**Target TRL**	At a minimum TRL8
IPR	CORONIS	**License**	FOSS such as GNU General Public License (GPL).
First availability	2020-08 Functionality showcase for tiled maps creation from 2D/2.5D maps, with software released in a container.		
MS4 expectation	First deployment within NEANIAS infrastructure, integrated with NEANIAS AAI, and also supporting 3D maps. (M14)		
MS6 expectation	Service integrated with all other core services of NEANIAS, first version of Cesium web app (M25)		
MS7 expectation	Final deployment of the tile maps creation service and the Cesium web app (M32)		

8.3. C4.2 Visualisation Gateway (VG)

C4.2 will provide a development environment for designing, rapid prototyping, implementing and fully testing complex visualisation solutions for realising common data exploration workflows. To serve as a universal core service for multiple users, the popular IPython based Jupyter Hub project has been selected. C4.2 will built upon this and the framework for visual discovery developed in C4.1. C4.2 services are envisaged to be interconnected with C3.1 to visualise AI powered solutions, C4.3 to underpin powerful VR/AR solutions and C4.4. to facilitate end-user data accessibility. C4.2 are expected to be deployed in a way that is fully embedded within the relevant workflows of end-user activities, while exploiting diverse parallelisation models and infrastructure accelerator capabilities.



C4.2 services diagram

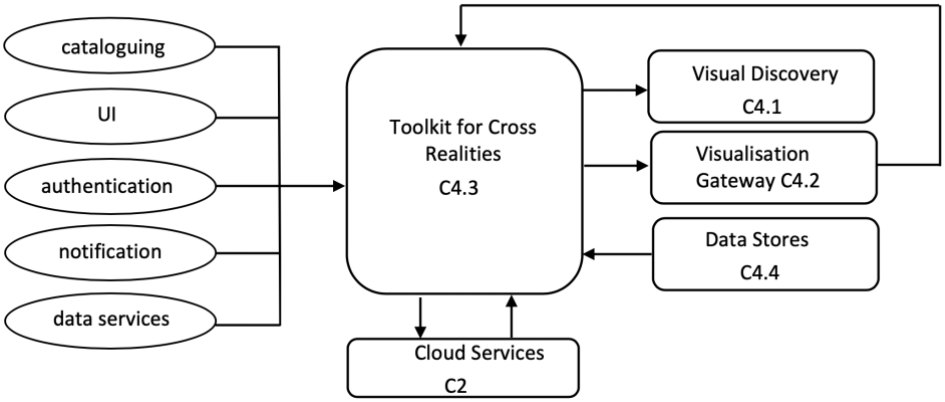
Short Name	VG (C4.2)	**Lead partner**	INAF
Type	Web/Service UI Web App	**Contributors**	JACOBS, UoP
Title	Visualization Gateway		
Master element	N/A		
Description	The service offers a web-based interactive environment for designing, developing and evaluating complex, scalable visualisation scenario solutions. Big data will be handled in a distributed environment leveraging on heterogeneous infrastructures and based on the services implemented in C4.1 while being interconnected with services in C4.3, C4.4. and C3.1 The visualisation outcomes will be strictly integrated within the overall scientific workflow seamlessly. Jupyter Hub was selected for		

	providing a multi-user development environment based on IPython, using a web interface for code implementation and interpretation.		
Technical details	The service offers an interactive framework, based on Jupyter Hub in connection with C3.1 AI gateway to enable efficient visualisation of complex visualisation tasks (including AI models) with outcomes being seamlessly integrated in the overall scientific workflows.		
Core integration	It may need the following integration: • Catalogue service (monitoring) • Common user interface components • AAI Authentication & Authorization Infrastructure Service • Notification service • Data services (depositing, sharing, transfer, publishing, exploration) • Computational resources access service		
Depends on	It may depend on: • C4.1.1 VD-VisIVO • C4.1.2 VD-Splotch • C4.1.3 VD-Maps • C4.3 XR • C4.4 DS • C3.1 AI gateway (Jupyter Hub technology)		
Use cases	a) NEANIAS space research services for structure detection with Machine Learning (S3) b) Use Cases from C4.1.1 (VD-VisIVO), C4.1.2 (VD-Splotch) and C4.1.3 (Maps)		
EOSC services integration	TBC		
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	At a minimum TRL8
IPR	INAF	**License**	FOSS such as GNU General Public License (GPL).
First availability	2020-08 planned month of first release with core visualisation Python packages to support NEANIAS outreach and engagement plan.		
MS4 expectation	First version for service showcasing (M14).		

MS6 expectation	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.
MS7 expectation	Final deployment (M32)

8.4. C4.3 Toolkit for Cross Realities (XR)

C4.3 will provide a toolkit underpinning an environment for designing, implementing and testing complex visualisation solutions exposed to end-users via Cross Realities (XR) mechanisms, e.g. those based on Virtual Reality (VR) and Augmented Reality (AR). To serve as a universal core service for multiple users, popular software frameworks and technologies have been selected based on a set of components used by an existing TLR6 software solution called Astro Data Navigator (ADN), e.g. the frameworks provided by the game engine Unity and HTC Vive headset. Such components will support interactive data exploration and navigation mechanisms e.g. for advanced comparisons in multidimensional and multifrequency datasets for research and public outreach. C4.3 services will be built upon extending components with a focus on providing services with enhanced realism, precision and usability along a number of aspects relating to enriched user experiences covering e.g. novel navigation mechanisms, advanced real-time tracking, VR/AR technologies and seamless integration of large-scale catalogues. C4.3 will exploit the framework for visual discovery developed in C4.1. C4.3 services are envisaged to be interconnected seamlessly with C4.2 to furnish complex visualisation workflows and C4.4 to underpin advanced data access. The components identified so far are a tracking system interface integrating with motion tracking, a data connector for retrieving, in a generic way, data coming from different sources (individual files or databases) and a positioning manager providing the capabilities to retrieve specific data about position/rotation of particular objects for specific temporal instants. The implementation of the later is envisaged to deploy cloud infrastructures to extract the relevant information through Spice kernels. As part of the XR toolkit various VR/AR viewers are expected to be realised visualising not only space objects but also 3D data of various typologies of interest to other NEANIAS domains (e.g. atmospheric).



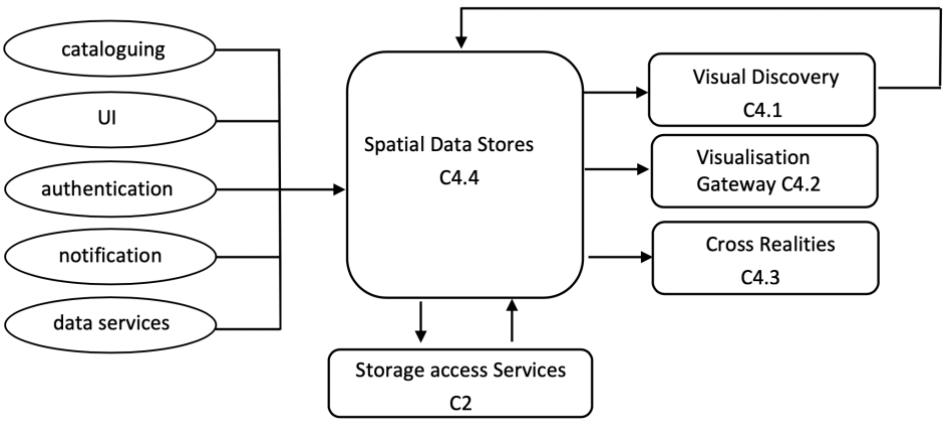
C4.3 services diagram

Short Name	XR (C4.3)	**Lead partner**	ALTEC
Type	Software / Web Service	**Contributors**	UoP
Title	Toolkit for Cross Realities (XR)		
Master element	N/A		
Description	The service offers a cloud enabled toolkit facilitating interactive data exploration and navigation to underpin VR/AR applications.		
Technical details	The service will be founded upon the Astro Data Navigator (ADN), which is a 3D visualization environment within the Unity game engine for large stellar catalogues. A number of services will be targeted for data pre-processing to prepare VR/AR enabled assets from a variety of sources focusing on realism, precision and usability of the relevant workflows and covering sophisticated navigation, real-time AR/VR tracking and seamless integration of large-scale catalogues.		
Core integration	It may need the following integration: • Catalogue service (monitoring) • Common user interface components • AAI Authentication & Authorization Infrastructure Service • Notification service • Data services (depositing, sharing, transfer, publishing, exploration) • Computational resources access service		
Depends on	It may depend on: • C4.1.1 VD-VisIVO		

	• C4.1.2 VD-Splotch • C4.1.3 VD-Maps • C4.4 DS		
Use cases	• Within NEANIAS, all space thematic services (S1, S2, S3). • Use cases of C4.1 services		
EOSC services integration	The service may be integrated with EOSC Cloud Container Computing services.		
EOSC Service	TBC		
Start TRL	TRL6	**Target TRL**	TRL8 as a minimum
IPR	ALTEC/UOP	**License**	FOSS such as GNU General Public License (GPL).
First availability	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan.		
MS4 expectation	First version for service showcasing (M14).		
MS6 expectation	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.		
MS7 expectation	Final deployment (M32).		

8.5. C4.4 Spatial Data Stores (DS)

Spatial Data Stores providing a proper set of reference systems and data structures useful to exploit the spatial features of catalogues and datasets.



C4.4 services diagram

Short Name	Spatial data stores (C4.4)	**Lead partner**	JACOBS
Type	DBMS	**Contributors**	ALTEC, INAF
Title	Spatial data stores		
Master element	N/A		
Description	Spatial data stores allow data to be referenced, query and retrieved, based on a given location (e.g. the globe, the Earth, planetary bodies with a proper reference system or the sky). Earth Observation data, including Copernicus datasets and products, are referenced in the ADAM Data Access System (DAS), a software module that manages a large variety of geospatial information that feature different data format, geographic / geometric and time resolution. It allows accessing, visualizing, sub-setting, combining, processing, downloading all data sources simultaneously. The DAS exposes OGC Open Search and Web Coverage Service (WCS 2.x) interfaces that allow discovering available datasets and subset them in any dimension with a single query. Planetary bodies differ from the sky on their concavity and dimensions, bodies being concave surfaces of limited sizes whereas the sky is convex and limitless. A coordinate reference system at the interface layer on spatial data bases is responsible to translate user requests in to internal data reference.		
Technical details	ADAM, Geoserver, MongoDB, PosgresSQL+PostGIS/PGSphere and and Rasdaman are systems designed to handle spatial data storage and the operations to query/retrieve such data sets. ADAM provides seamless full data cycle management functionalities to explore spatial distribution and temporal evolution of various geophysical and geospatial information as well as to integrate and execute data processing functionalities at scale. Rasdaman is designed to handle raster data, it specifically works with spatial arrays of two or more dimensions. MongoDB, on the other hand, is designed to work with unstructured data, text, numbers and vectors, it has applicability on spatial vector data -- data representing polygons used to define observation footprints, for example.		

	Geoserver can handle both kinds of data, raster and vectors, it is a database system that offers wider applicability and simpler implementation. PostgreSQL+PostGIS/pgSphere. PostGIS provides support for geographic objects allowing location queries to be run in SQL and follows the Open Geospatial Consortium's "Simple Features for SQL Specification" and has been certified as compliant with the "Types and Functions" profile. Furthermore, PostGIS enjoys wide support from various third-party open source and proprietary tools as QGIS. pgSphere, instead, is an extension for spherical geometry allowing fast research in astronomical or geographical application. The advantage of Rasdaman and MongoDB relies on performance. MongoDB is designed for horizontal scalability -- i.e., the system naturally works in a distributed environment, in parallel -- making it a promising option for EOSC/NEANIAS. The open, community version of Rasdaman does not work in parallel but offers top performance and very little memory footprint which makes it a stable option for highly demanding services. Geoserver comes as a good alternative for small size, heterogeneous data sets, as a volatile data storage system to publish high-level or temporary user products.		
Core integration	a. Authentication and Authorization b. Storage access services		
Depends on	• Data exploration services		
Use cases	a) Atmospheric (Ax) and Space research services (S1) to manage space data b) Provide OGC compliant services for data access (Ax, S1, S2)		
EOSC services integration			
EOSC Service	No		
Start TRL	TRL6	**Target TRL**	TRL8
IPR	JACOBS MEEO	**License**	GPL v3.0 proprietary software
First availability	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan.		
MS4 expectation	Service setup / First version for showcasing (M14)		
MS6 expectation	Integrated showcasing (M25)		

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

MS7 expectation	Final deployment (M32)

9. Workplan – Timeline

The following section will overview the four stages of each service during the lifetime of the project. The four stages will represent four milestones during the development and delivery of the services. The table below has the four stages/ milestones in columns for which each service defines its expected status. The evolution of the services starts from standalone service till the fully integrated version.

Service Title	First Availability	MS4 Expectation (M14)	MS6 Expectation (M25)	MS7 Expectation (M32)
Catalogue Service	2020-08 Planned month of preliminary release to support NEANIAS outreach and engagement plan and brief reference of what may be usable at the point. No significant integration is expected.	Prototype deployment. Not populated with NEANIAS service metadata. EOSC integration is not mandatory for EOSC services yet.	Deployed and populated with metadata of all NEANIAS services.	Issue fixing and improvements.
Research Product Catalogue	Already available			
Data Validation Service	2020-08 First version deployed with a local database, REST interface only, provided to support integration with other services.	Prototype deployment, REST interface operational, without automated rating, integrated with NEANIAS AAI.	Both Web UI and REST interfaces deployed and operational, without automated rating.	If deemed publishable to EOSC, it will be fully integrated to EOSC at this point. Additionally, if automated rating proves viable, it will be operational at this point.

NEANIAS is funded by European Union under Horizon 2020 research and innovation programme via grant agreement No. 863448

Web UI toolkit and template	2020-05 Generic stylesheet and color guidelines.	Generic components for information presentation	Specific components for service exposure. Additional UI elements, such as buttons and menus.	Fine-tuned UI elements.
Catalogue Portal	2020-08. Planned month of preliminary release to support NEANIAS outreach and engagement plan and brief reference of what may be usable at the point. No significant integration is expected.	Prototype deployment. Not populated with NEANIAS service metadata. EOSC integration is not mandatory for EOSC services yet.	Deployed and populated with all NEANIAS service metadata.	Issue fixing and improvements.
OpenDMP Software	2020-02 Public release to accommodate observations of NEANIAS users. Deployed on OpenAIRE as Argos.	Integration with EOSC AAI.	Integration with PID providers.	Final release.
Data Publishing service	Already available			
Persistent Identifier Service	Already available			

Authentication & Authorization Infrastructure Service	2020-04: At this preliminary release, it is expected that at a minimum the dev and staging realms will be supported. Role based authorization will be available at the realms and at the client level. Federated users will be limited to NEANIAS AAI	Production realm will be made available. Alpha integration with NEANIAS Logging and NEANIAS Accounting services will be available.	EOSC AAI integration will be available. Enhancements on the integration with the NEANIAS Accounting Service. Alpha availability of fine-grained resource level authorization will be evaluated.	Final version of selected fine-grained authorization capabilities, documentation, Accounting and Logging integration
Configuration Management Service	2020-07 Initial service delivery dependable for services to store and retrieve public configuration	Major service delivery, integrated with AAI.	Major service update supporting secrets.	Minor fixes and updates if required – no functional changes
Service Instance Registry	2020-07 Initial service delivery dependable for services to store and retrieve targeted service entries	Major service delivery, integrated with AAI with preliminary service instance model.	Major service with final service model.	Minor fixes and updates if required – no functional changes
Log Aggregation Service	2020-06: At this preliminary release, it is expected that basic configuration will be available to allow for direct aggregation of messages. No user interface to browse the logs will be available	File and http beats configuration to support distributed log aggregation, basic transformations to extract key characteristics	Graphical user interface to browse and analyze log entries through a Kibana interface with direct access to the underlying index	Integration with the NEANAIS AAI to facilitate federated user role based mapping to visualization and possibly administrative operations

Accounting Service	2020-10 At this preliminary release, it is expected that basic ability to register accounting information according to NEANIAS accounting model will be provided. Visualisation of the information will be at the level of record.	Refinements on common accounting models. Aggregation utilities to ship data efficiently. Basic user interface without reporting functionality	Extended accounting model and basic reporting functionality	Full accounting model and reporting functionality available
Notification Service	2020-07: Initial version offering single channel communication. Integration entry points through subset of available channels.	Extended notification templates. User profiling endpoints and notification preferences. Notification context	Additional notification channels including mail and in-app notifications. Web user interface widget for in-app notification browsing	User profile integrations for automated profiling
Object Storage Service	2019-11 The service is already available at the GARR Cloud Platform premises https://cloud.garr.it	Service is present. If required configuration/allocation changes may occur.	Service is present. If required configuration/allocation changes may occur.	Service is present. If required configuration/allocation changes may occur.
Data Sharing Service	2020-08 First version deployed with a local database, provided to support integration with other services.	Prototype deployment, integrated with NEANIAS AAI.	Deployed and operational.	If deemed publishable to EOSC, it will be fully integrated to EOSC at this point.

Data Transfer service	2020-08 Deployed on NEANIAS infrastructure. The software is already available at https://github.com/SZTAKI-LPDS/data-avenue	Initial deployment on GARR cloud and integration to Swift object store on GARR cloud happens.	Integration to some of the core services such as AAI, Logging is available in the service.	Final deployment; Integration to further core services such as Accounting, Notification and to all other necessary services. EOSC integration happens.
Data exploration service	2020-08 Initial deployment on NEANIAS infrastructure, with a set of data querable (e.g. Copernicus datasets)	Service setup, including integration with NEANIAS AAI component. Registration in the central service repository	Showcase of data exploration. Logging available	Final deployment
GARR Cloud Platform Service	The service is already available at the GARR Cloud premises: https://cloud.garr.it	-	-	-
GARR Container Platform Service	The service is already available at the GARR Cloud premises: https://cloud.garr.it/containers/	-	-	-
GARR Deployment as a Service	The service is already available at the GARR Cloud premises: https://cloud.garr.it/apps/daas/	-	-	-

MiCADO Deployment as a Service	2020-08 Deployed on NEANIAS infrastructure. The service is already available at https://micado-scale.eu/	Initial deployment; MiCADO is deployed, several examples are developed in order to demonstrate how MiCADO can be used to automatically deploy and scale reference architecture	Integrated showcasing; Integration to some of the core services such as AAI, Logging is available in the service	Final deployment; Integration to further core services such as Accounting, Notification and to all other necessary services is available in MiCADO. EOSC integration happens.
AI Science Gateway: service for Development of Machine Learning Model using Jupyter Hub	2020-06, initial version with core Python packages	Prototype deployment for showcasing; basic IDE functionality	Integrated showcasing; integration of some core services	Final deployment
Serving trained ML models	2020-09, initial version with core model serving capabilities based on open source solution.	Prototype deployment for showcasing; experiment level.	Integrated showcasing; integration of some of the core services.	Final deployment
Distributed Multi-GPU training of large ML models using Horovod	2020-08, initial version	Prototype deployment for showcasing; experiment level	Integrated showcasing; integration of some of the core services	Final deployment

Distributed Machine Learning using SparkML	2020-08, initial version	Prototype deployment for showcasing; experiment level	Integrated showcasing; integration of some of the core services	Final deployment
Visual Discovery Framework - VisIVO	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan. Release of the software as Docker containers.	First version for service showcasing deployed within NEANIAS infrastructure(M14).	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.	Final deployment (M32).
Visual Discovery Framework - Splotch	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan. Release of the software as Docker containers.	First version for service showcasing deployed within NEANIAS infrastructure (M14).	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.	Final deployment (M32).
Visual Discovery Framework - 2D/3D Tiling and Maps	2020-08 Functionality showcase for tiled maps creation from 2D/2.5D maps, with software released in a container.	First deployment within NEANIAS infrastructure, integrated with NEANIAS AAI, and also supporting 3D maps. (M14)	Service integrated with all other core services of NEANIAS, first version of Cesium web app (M25)	Final deployment of the tile maps creation service and the Cesium web app (M32)

Visualization Gateway	2020-08 planned month of first release with core visualisation Python packages to support NEANIAS outreach and engagement plan.	First version for service showcasing (M14).	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.	Final deployment (M32)
Toolkit for Cross Realities (XR)	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan.	First version for service showcasing (M14).	Integrated showcasing (M25). Updates of software to provide integration with other NEANIAS core/thematic services based on use cases requirements updates.	Final deployment (M32).
Spatial data stores	2020-08 planned month of preliminary release to support NEANIAS outreach and engagement plan.	Service setup / First version for showcasing (M14)	Integrated showcasing (M25)	Final deployment (M32)

10. Conclusions

This deliverable aims a) to summarise the design principles for implementing the NEANIAS core services b) to overview the architecture which will provide a framework for combining the core services and c) to introduce the C1/C2/C3/C4 core services with their most important attributes to be developed in the rest of the project.

Section on design principles is focusing on service-oriented architecture with high-degree of interoperability through the utilisation of REST paradigm and standards in communication and FAIR principles in data handling. Guidelines for operating services and for the most important security aspects are also collected. The AAI security concept and policy have been introduced in details since one of the most commonly used function in the core services will be authentication/ authorisation/ identification. Similar importance is given for user interface design where basic principles are declared. Finally, the most recommended technologies are overviewed on the different fields such as programming languages, development/data management/AI and other frameworks.

Section on system architecture is giving a detailed overview of the NEANIAS ecosystem being developed within the framework of the NEANIAS project. The deliverable provides a top-to-bottom overview. It starts with a high-level architecture diagram summarising the abstract relation among the Basic / Advance core services and Thematical ones. Then a logical architecture view with the fundamental building blocks is presented by specifying 5 levels of abstraction. In order to support the common internal operation of the core services, an internal lifecycle is proposed. Finally, the hypothetical deployment of the NEANIAS services is presented to help the reader understand the overall architecture of the ecosystem.

Fundamental resources abstractions are specified and detailed by introducing three categories, such as storage, compute and other. These sections summarise the identified resources to be utilised for testing, development and operation of the services introduced in the next section of the deliverable.

In the next 4 sections, the list of C1/C2/C3/C4 services are specified with their most important attributes. C1 services provide the necessary tools for NEANIAS services to be discoverable and accessible and integrated with EOSC hub. C2 services form the lower level of services in the NEANIAS ecosystem and are the ones that deliver access to various levels of resources that serve the use cases of the project. C3 services provide features of a typical Machine Learning (ML) workflow lifecycle, while C4 services provide facilities that may reach up to the end user offering features of a typical visualisation workflow lifecycle.

Finally, a workplan is given for overviewing the stages of the services evolving during the project. From standalone running to different levels of integration the services will evolve during the project through 4 milestones.

NEANIAS is funded by European Union under Horizon 2020 research and innovation programme via grant agreement No. 863448

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

List of acronyms

Acronym	Description
AAI	Authentication Authorization Infrastructure
ACL	Access Control List
API	Application Programming Interface
CSS	Cascading Style Sheet
DMP	Data Management Plan
EOSC	European Open Science Cloud
FAIR	Findable, Accessible, Interoperable, and Reusable
FOSS	Free and Open Source Software
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
JS	JavaScript
JVM	Java Virtual Machine
MVC	Model-View-Controller
ReST	Representational State Transfer
RI	Research Infrastructure
SOA	Service Oriented Architecture
SKOS	Simple Knowledge Organization System
SPA	Single Page Application
VisIVO	Visualization Interface for the Virtual Observatory

Appendix 1 – Service Description Template

Short Name	Short name for the service	**Lead partner**	Partner responsible as per DOA or post-DOA agreement
Type	Software library / Web service / UI web app / UI Component	**Contributors**	Partners contributing to the service
Title	Full element name		
Master element	Service this element belongs to or specializes or extends (if applicable)		
Description	Brief description of the element oriented to the general public covering objectives and main functional outcome.		
Technical details	Technical details about the service, w.r.t. techniques, protocols etc Also describe whether the element is web-ui accessible or a background service.		
Core integration	Describe integration of the service with fundamental core services of NEANIAS, e.g. - Authentication / Authorisation - Accounting - Logging - Data discovery / Data publication - Storage access (temporary / permanent) - Computation access (direct or indirect) - Visualisation - other		
Depends on	Other services or components the element depends on, either in NEANIAS or other infrastructures.		
Use cases	Use cases depending on the element (coming from thematic sectors and business sectors, or fundamental requirements of NEANIAS infrastructure)		
EOSC services integration	Services from EOSC the service may expect to use or integrate with		
EOSC Service	Is published in EOSC ? (Yes/No)		
Start TRL	TRL6/TRL7/TRL8/TRL9	**Target TRL**	TRL8/TRL9
IPR	Name the main IPR holder of the service (expected to	**License**	Name the license under which the code

Core Services Architecture, Design Principles and SpecificationsCore Services Architecture, Design Principles and Specifications

	coincide with partner responsible).		of the element may be delivered
First availability	yyyy-mm (planned month of preliminary release to support NEANIAS outreach and engagement plan) and brief reference of what may be usable at the point (no significant integration is expected)		
MS4 expectation	Describe expected status of the service at MS4, Prototype deployment for showcasing (M14) (EOSC integration is not mandatory for EOSC services yet)		
MS6 expectation	Describe expected status of the service at MS6, MS6 Integrated showcasing (M25)		
MS7 expectation	Describe expected status of the service at MS7, final deployment (M32) (EOSC integration is mandatory for services targeting EOSC)		