



nffa.eu

PILOT 2021
2026

DELIVERABLE REPORT

WP9 VA – Virtual Access

D9.1

Deployment of the VA service prototypes

Due date

M13



This initiative has received funding from the EU's H2020 framework program for research and innovation under grant agreement n. 101007417, NFFA-Europe Pilot Project

PROJECT DETAILS

PROJECT ACRONYM

NEP

PROJECT TITLE

Nanoscience Foundries and Fine Analysis - Europe|PILOT

GRANT AGREEMENT NO:

101007417

FUNDING SCHEME

RIA - Research and Innovation action

START DATE

01/03/2021

WORK PACKAGE DETAILS

WORK PACKAGE ID

WP9

WORK PACKAGE TITLE

VA – Virtual Access

WORK PACKAGE LEADER

Dr. Rossella Aversa (KIT)

DELIVERABLE DETAILS

DELIVERABLE ID

D – D9.1

DELIVERABLE TITLE

Deployment of the VA service prototypes

DELIVERABLE DESCRIPTION

This Deliverable presents the prototype of the first Virtual Access service: an instance of the MetaStore, named MetaRepo. The service has been installed as a virtual machine hosted at KIT, and has been connected to the Single-Sign-On. The service can be accessed using a Graphical User Interface. A monitoring service on the MetaRepo side makes a call to the API of the database test instance at each action performed in the MetaRepo by logged-in users, in order to keep track of the activities of the Virtual Access service.

DUE DATE

M13 31/03/2022

ACTUAL SUBMISSION DATE

DD/MM/YYYY

AUTHORS

Rossella Aversa (KIT)
Giuseppe Piero Brandino (eXact Lab)



Sabrina Chelbi (KIT)
Volker Hartmann (KIT)
Thomas Jejkal (KIT)

PERSON RESPONSIBLE FOR THE DELIVERABLE

Dr. Giuseppe Piero Brandino (eXact Lab)

NATURE

- R - Report
- P - Prototype
- DEC - Websites, Patent filing, Press & media actions, Videos, etc
- O - Other

DISSEMINATION LEVEL

- P - Public
- PP - Restricted to other programme participants & EC: (Specify)
- RE - Restricted to a group (Specify)
- CO - Confidential, only for members of the consortium



REPORT DETAILS

ACTUAL SUBMISSION DATE

DD/MM/YYYY

NUMBER OF PAGES

23 (right-click and select "update the field")

FOR MORE INFO PLEASE CONTACT

Dr. Giuseppe Piero Brandino
eXact Lab
Via Crispi, 56 – 34126 Trieste
(Italy)

email: brandino@exact-lab.it

VERSION	DATE	AUTHOR(S)	DESCRIPTION / REASON FOR MODIFICATION	STATUS
1	24/02/2022	R. Aversa	First draft	Draft
2	28/02/2022	R. Aversa	Introduction	Choose an item.
3	02/03/2022	R. Aversa, T. Jejkal	Naming convention, Access to Resources	Choose an item.
4	09/03/2022	V. Hartmann, R. Aversa	MetaRepo, Access monitoring	Choose an item.
5	09/03/2022	S. Chelbi	MetaRepo GUI	Choose an item.
6	11/03/2022	G.P. Brandino	SSO	Choose an item.
7	13/03/2022	R. Aversa	Revision, References	Choose an item.
8	18/03/2022	R. Aversa	Major Revision of naming convention	
9	29/03/2022	All	Final Version	

CONTENTS

Introduction	6
1. MetaRepo	6
2. MetaRepo GUI	8



2.1 Schema Management	8
2.2 Metadata Management	14
3. Naming convention	18
4. MetaRepo Authorization and Authentication	21
4.1 Single Sign On (SSO)	21
4.2 Access to Resources	21
5. MetaRepo Access Monitoring	22
References	23



INTRODUCTION

Virtual Access (VA) is a novel service offered in NEP. It comprises innovative online simulation services, databases, machine learning services, data and metadata services, which will be seamlessly integrated into the NEP infrastructure and provided as cloud services to authenticated users. The access will be provided by the facilities hosting the services. Each Access Provider will operate its own infrastructure, made of one or more installations. All involved Access Providers will offer identified User Access to the VA services under their own schemes based on institutional policies.

Part of the online data services that will be made available as VA are currently (or will be) developed in the Joint Activity (JA) 6 on FAIR data approach. According to the NEP proposal, seven services are planned, even though scouting activities to enlarge the offer are foreseen. The services are at different stages of development, and will be gradually completed and integrated into the offer.

This deliverable presents the first prototype of a VA service: the MetaRepo, a metadata repository to register and validate Metadata Schemas and Metadata Documents. The document is structured as follows: Section 1 describes the MetaRepo service, and Section 2 shows the functionalities of its Graphical User Interface (GUI). Being the purpose of this Section only demonstrative, for the sake of simplicity the functionalities of the MetaRepo GUI are shown on basic examples instead of being applied to a real NEP use case. This choice has been made to prevent the confusion which might arise due to the complexity of the Metadata Schemas. A real usage example is presented in Milestone 10. Section 3 shows how the Authorization and Authentication has been implemented for the MetaRepo. Section 4 explains how the monitoring of user accesses and actions is performed.

1. MetaRepo

MetaRepo is the NEP instance of MetaStore [1]. It offers a generic metadata repository and Metadata Schema registry which provides metadata versioning and, at later time, metadata search and visualisation. From the technological perspective, it will be accessible via RESTful service interfaces. It enables Data Curators to register Metadata Schemas in one of the two supported formats: XML Schema Definition (XSD) or JSON Schema, and it allows Research Users to store Metadata Documents which are automatically validated at upload time against the corresponding registered Metadata Schema.

A Metadata Schema describes the structure and the content of the XML or JSON Object, which should be registered in the MetaRepo.

While registering a Metadata Schema, a Metadata Schema Record is required. It represents a JSON document and includes administrative metadata of the Metadata Schema (e.g. Creator, Creation Date, Access Rights).

A Metadata Document describes an XML or JSON resource, which should be ingested in the MetaRepo.

While registering a Metadata Document, a Metadata Document Record is required. It is a JSON document and includes administrative metadata of the Metadata Document.

The MetaRepo will be populated with Metadata Documents linked to scientific Datasets resulting from Transnational Access Experiments, as well as with administrative Metadata Documents regarding the related Proposal and the Research User information. This additional metadata in a standardised format, i.e. DataCite [2], provides further contextualisation of each Dataset besides the technical description of the Dataset itself, which serves further applications of metadata, such as search or publication.

As stated in the NEP Data Management Plan (DMP) [3], Research Users must ensure Open Access (free of charge online access for any user) to all Publication Data, that is the Research Data generated in the NEP project needed to validate the Results presented in a Scientific Publication or appearing in it, including all the relevant metadata.

At publication time, the Research User will not need to collect and check the metadata: using the MetaRepo at each step of the Study (e.g., Sample Preparation, Measurement, Data Analysis) will ensure to already have complete validated metadata, linked to the relevant Raw and/or Analysed Data.

The first version of the prototype is available at [4] and can be accessed thanks to its GUI (see Section 2). It runs on a virtual machine (4 CPUs, 16 GB RAM, 50 GB SDD, OS: Debian 11 (Bullseye), SSL via Apache reverse proxy) hosted at KIT. Currently, only authenticated users are allowed to read/write documents based on access control lists. A more detailed description of how the access to resources is managed can be found in Section 4.2.

Authentication is done via SSO (described in Section 4.1), which is provided by the NFFA-Europe portal [5]. Every access to the MetaRepo that requires to read, write or modify Metadata Resources, as well as to retrieve relevant information, such as Research User and Proposal Data, is authenticated. After authentication, Research Users can browse all the available Metadata Schemas and Documents according to the access control list (Section 4.2), or to search, retrieve, add and edit the Metadata Document describing a specific Measurement.

Once the Research Users have selected an appropriate Metadata Schema for their Research Data, they can create a Metadata Document and store it in the MetaRepo. Each Metadata Document must be associated with a registered version of a Metadata Schema and its data resource, in order to be validated during the ingest process. In case of successful validation, the Metadata Document is saved and associated to a unique URL; otherwise, the ingest is rejected. Each valid update of a Metadata Document results in a new version of the document, accessible/referenceable via a unique URL.

In future releases, the different Metadata Documents will be co-referenced using internal IDs. In this way, the MetaRepo will give the possibility to match Proposals with publications and the corresponding Data Provenance, offering a clearer insight on the scientific process from the project idea to the research output. Currently, a naming convention for Metadata Record IDs has been established (see Section 3) to keep track of the Data Provenance.

After a testing and hardening phase, the MetaRepo is expected to be available as a prototype with full functionalities at Month 25, and will be in production state during the last three years of the NEP project. The Deliverable 9.2, due at Month 31, will include the additional features of the next versions of the MetaRepo prototype.

2. MetaRepo GUI

The MetaRepo GUI, accessible at [4] offers an intuitive user interface, which allows the easy use of the MetaRepo. This user-friendly GUI has been based on the Metadata Editor [6], a JavaScript Library developed at the Karlsruhe Institute of Technology - Steinbuch Centre for Computing (KIT-SCC), which has been integrated in the MetaRepo. With the help of the GUI, the Research User can perform all the RESTful operations provided by the service on Metadata Schemas and on Metadata Documents, which will be shown in Sections 2.1 and 2.2, respectively.

The GUI provides a dashboard page, shown in Figure 1, which includes two large buttons "Schema Management" and "Metadata Management".

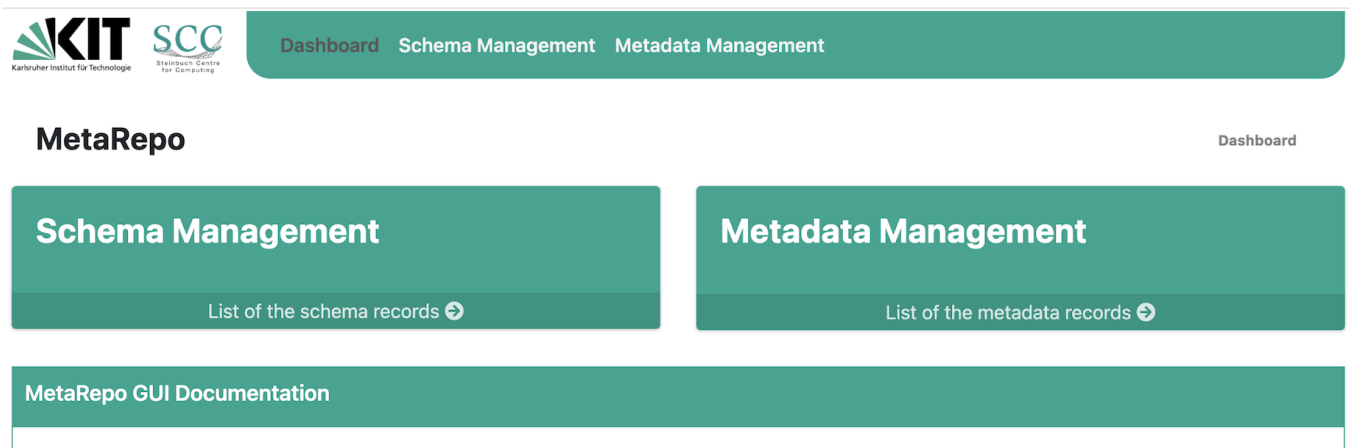


Figure 1: Dashboard Page

With the help of these buttons, all stored Metadata Schema Records and Metadata Document Records can be listed. At the bottom of the page, a GUI documentation will be added to give more information on how to use the GUI. At the top of the page, a menu allows users to easily navigate through the available pages, which are "Dashboard", "Schema Management" (explained in Section 2.1) and "Metadata Management" (explained in Section 2.2). To facilitate the navigation, the name of the current page is coloured in dark grey in the menu, and it also appears on the right side at the bottom of the menu.

2.1 Schema Management

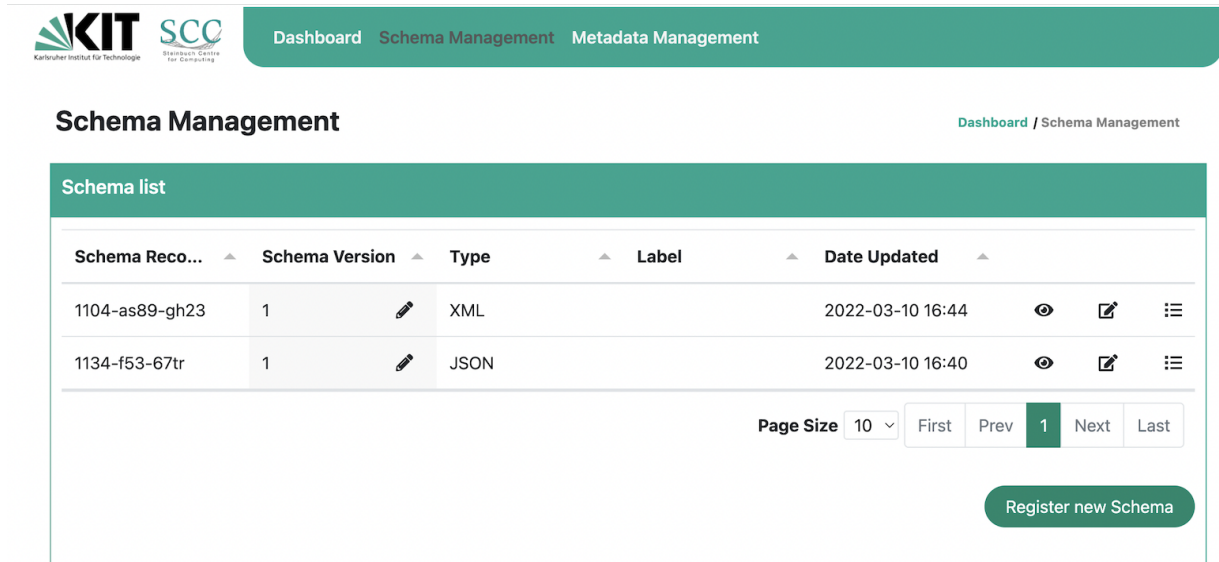
The actions offered to the users in the "Schema Management" page are:

- A. To list all the available Metadata Schema Records
- B. To Create a new Metadata Schema Record in order to register a new Metadata Schema
- C. To visualise a Metadata Schema Record
- D. To visualise a Metadata Schema
- E. To update a Metadata Schema Record and/or a Metadata Schema
- F. To list only the available Metadata Document Records related to a given Metadata Schema

The user can list all the available Metadata Schema Records (action A) via the large button or the menu item "Schema Management". The page includes a table, shown in Figure 2, which lists all registered Metadata Schema Records. The table header includes the most important properties of

the Record. The user can choose the pagination size (i.e., the number of Metadata Schema Records listed in each page) of the table by changing the page size at the bottom.

As a Metadata Schema can have different versions, the user can visualise and update a specific Version of the Metadata Schema, or list the Metadata Document Records based on a specific Schema Version. By clicking on the pencil icon in the Schema Version (shown in Figure 2), the user can select the Metadata Schema Version.



The screenshot shows a web interface for 'Schema Management'. At the top, there are logos for KIT (Karlsruher Institut für Technologie) and SCC (Strategic Centre for Computing). The navigation bar includes 'Dashboard', 'Schema Management', and 'Metadata Management'. The main heading is 'Schema Management' with a breadcrumb trail 'Dashboard / Schema Management'. Below this is a 'Schema list' section containing a table with the following data:

Schema Reco...	Schema Version	Type	Label	Date Updated			
1104-as89-gh23	1	XML		2022-03-10 16:44			
1134-f53-67tr	1	JSON		2022-03-10 16:40			

Below the table, there is a pagination control with 'Page Size' set to 10, and buttons for 'First', 'Prev', '1', 'Next', and 'Last'. A 'Register new Schema' button is located at the bottom right of the table area.

Figure 2: List of registered Metadata Schema Records (action A). The table includes Metadata Schemas in the two supported formats: JSON and XML

To register a new Metadata Schema (action B), the "Register new Schema" button can be used. This action triggers the generation of a form (shown in Figure 3), which allows the user to provide the administrative metadata necessary to manage the Metadata Schema itself (i.e., to create the Metadata Schema Record). In the same form, the user can upload the Metadata Schema and validate the inputs.

Figure 3: Creation of a Metadata Schema Record for a new Metadata Schema (action B)

A message is shown to notify whether the operation is successfully executed or not. In case the Metadata Schema Record is created, it is listed in the table. As shown in Figure 2, different actions are possible on registered resources, represented by icons: to visualise the properties of the Metadata Schema Record (action C, “eye icon”), to update some fields of the Metadata Schema Record and/or the Metadata Schema (action E, “edit icon”) and to list of all the ingested Metadata Document Records validated against the given Metadata Schema (action F, “list icon”). The user can move the mouse over the icons to read their functionalities.

By clicking on the “eye icon” (action C), all the properties of the Metadata Schema Record are visualised in a form, shown in Figure 4.

Schema Record Form [Close]

Schema Record Identifier: 1104-as89-gh23

Schema Version: 1

mime Type: application/xml

Type: XML

Date Created: 2022-03-10 16:44

Date Updated: 2022-03-10 16:44

Label: [Empty]

Definition: [Empty]

Comment: [Empty]

ACL

Permission: ADMINISTRATE

sid: SELF

[Add] [Delete]

Locked: [Unlocked]

Schema Document Uri: http://localhost:8040/api/v1/schemas/1104-as89-gh23?versi

Schema Hash: sha1:5f1e5cbf32bf15c4931af19a49ea35c5c28f5c7b

[Schema]

Figure 4: Visualisation of the properties of the Metadata Schema Record (action C)

The form includes, in turn, another button called “Schema” in order to visualise the Metadata Schema registered in the Record (action D), which generates the Metadata Schema as a form in case the file format is JSON (Figure 5), or simply shows the Metadata Schema if the file format is XSD (Figure 6).

Schema Record Form [Close]

Title

[Empty Input Field]

Title of object.

Name

[Empty Input Field]

Name of object.

Figure 5: Visualisation of the JSON Metadata Schema Form (action D)

Schema Record Form
✕

XSD Schema

```

<xs:schema xmlns="http://www.example.org/schema/xsd/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/schema/xsd/"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<xs:element name="metadata">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Figure 6: Visualisation of the XSD Metadata Schema Form (action D)

By clicking on the “edit icon”, the user can update the Metadata Schema Record and/or the Metadata Schema (action E). As illustrated in Figure 7, the Metadata Schema can be updated either manually by using the generated form, or by uploading a new file. In case no file is uploaded, only the Metadata Schema Record is updated.

Schema Record Form
✕

Schema Record Identifier*

mime Type

Type

ACL

sid

Permission

Add
Delete

Locked

file Schema

updated-meta...ta-schema.json

Update Schema

Figure 7: Update of the Metadata Schema Record and the Metadata Schema (action E)

By clicking on the “list icon”, the user can list all Metadata Document Records related to a given Metadata Schema (action F), as shown in Figure 8.

Dashboard Schema Management **Metadata Management**

Metadata Management

Dashboard / Metadata Management / 1134-f53-67tr

Metadata list

Identifier	Related Resource	Schema Identifier	Date Updated	
89ad2b09-d042-4ed3-bb...	resource_1134	http://localhost:8040/api/v...	2022-03-11 09:16	👁 ✎

Page Size

First
Prev
1
Next
Last

Register new Metadata Document

Figure 8: List of Metadata Document Records related to the Metadata Schema “1134-f53-67tr” (action F)

2.2 Metadata Management

The same actions described in Section 2.1 for Metadata Schemas can be executed on Metadata Documents in the “Metadata Management” page, where the same page layout is used. In particular:

- G. To list all the available Metadata Document Records
- H. To Create a new Metadata Document Record in order to register a new Metadata Document
- I. To visualise a Metadata Document Record
- J. To visualise a Metadata Document
- K. To update a Metadata Document Record and/or a Metadata Document

The user can list all the Metadata Document Records ingested in the MetaRepo (action G) via the large button or the menu item “Metadata Management”. The page includes a table, shown in Figure 9, which lists all the registered Metadata Document Records. The table header includes the most important properties of the Record, in particular the identifier of the Metadata Schema it is based on. The user can choose the pagination size (i.e., the number of Metadata Document Records listed in each page) of the table by changing the page size at the bottom.

The screenshot shows the 'Metadata Management' page. At the top, there are logos for KIT (Karlsruher Institut für Technologie) and SCC (Steinbuch Centre for Computing). A navigation bar contains 'Dashboard', 'Schema Management', and 'Metadata Management'. The main heading is 'Metadata Management' with a breadcrumb 'Dashboard / Metadata Management'. Below this is a 'Metadata list' table with the following data:

Identifier	Related Resource	Schema Identifier	Date Updated		
6aa4f15f-c74c-42e4-b6bb...	resource_1104	http://localhost:8040/api/v1...	2022-03-11 09:18		
89ad2b09-d042-4ed3-bb...	resource_1134	http://localhost:8040/api/v1...	2022-03-11 09:16		

Below the table, there is a 'Page Size' dropdown set to '10', and navigation buttons for 'First', 'Prev', '1', 'Next', and 'Last'. A green button labeled 'Register new Metadata Document' is located at the bottom right of the table area.

Figure 9: List of all registered Metadata Document Records (action G). The table includes two Metadata Documents based on different Metadata Schemas

To create a new Metadata Document Record in order to register a new Metadata Document (action H), the button “Register new Metadata Document” can be used. This action triggers the generation of a form (shown in Figure 10) which allows the user to provide the administrative metadata necessary to manage the Metadata Document itself (i.e., to create the Metadata Document Record). It is important to remark that a Metadata Document must always be validated against a registered Metadata Schema; thus, the “Schema” field is mandatory.

In the same form, the empty Metadata Document can be generated according to the selected Metadata Schema and the form can be filled in manually, or the Metadata Document can be uploaded as a file in any supported format (JSON or XML). If the document is uploaded, a filled

form is generated in case of JSON format (as shown in Figure 12) or the file is visualised in case of XML format (as shown in Figure 13), so that the inputs can be checked by the user before registering it.

The screenshot shows a 'Metadata Record Form' with the following sections and fields:

- Identifier:** A text input field containing 'record_1134'.
- Related Resource:**
 - Identifier*:** A text input field containing 'resource_1134'.
 - Identifier Type*:** A text input field containing 'INTERNAL'.
- Schema:**
 - Identifier*:** A text input field containing '1134-f53-67tr'.
 - Identifier Type*:** A dropdown menu with 'INTERNAL' selected.
 - Schema Version:** A text input field containing '1'.
- ACL:**
 - Sid:** A text input field containing 'SELF'.
 - Permission:** A dropdown menu with 'ADMINISTRATE' selected.
- Metadata Document:**
 - Choose file:** A button next to the text 'metadata-document.json'.
 - Show Input Form:** A green button.

At the top right of the form, there are two green buttons: 'Add' and 'Delete'.

Figure 10: Creation of a Metadata Document Record for a new Metadata Document (action H)

In case the Metadata Document is ingested correctly, it is listed in the table containing all the available Metadata Document Records (action G). As shown in Figure 9, different actions are possible on registered resources, represented by icons: to visualise the properties of the Metadata Document Record (action I, "eye icon") and to update some fields of the Metadata Document Record and/or the Metadata Document (action K, "edit icon"). The user can move the mouse over the icons to read their functionalities.

By clicking on the "eye icon" (action I), all the properties of the Metadata Document Record are visualised in a form, shown in Figure 11.

Metadata Record Form
✕

Identifier

Related Resource

Identifier

Identifier Type

Schema

Identifier

Identifier Type

Record Version

Date Created

Date Updated

Schema Version

ACL

Permission

Sid

Metadata Document URI

Document Hash

Figure 11: Visualisation of the properties of the Metadata Document Record (action I)

The form includes, in turn, another button called “Metadata Document” in order to visualise the Metadata Document registered in the Record (action J), which generates the Metadata Document as a form in case the file format is JSON (Figure 12), or simply shows the Metadata Document if the file format is XML (Figure 13).

Metadata Record Form
✕

Title

Title of object.

Name

Name of object.

Figure 12: Visualisation of the JSON Metadata Schema Form (action J)

Metadata Record Form ✕

XML Metadata Document

```

<?xml version='1.0' encoding='utf-8'?>
  <example:metadata
xmlns:example="http://www.example.org/schema/xsd/" >
  <example:name>My first XML document</example:name>
</example:metadata>
```

Figure 13: Visualisation of the XML Metadata Document (action J)

By clicking on the “edit icon”, the user can update some properties of the Metadata Document Record and/or the Metadata Document (action K). As illustrated in Figure 14, the Metadata Document can be updated either manually by using the generated form (shown in Figures 12 and 13), or by uploading a new file. In case no document is uploaded, only the Metadata Document Record is updated.

Metadata Record Form
✕

Identifier

Related Resource

Identifier

Identifier Type

ACL

Sid

Permission

Add
Delete

Metadata Document File

Choose file
No file chosen

Show Metadata Document

Figure 14: Update of the Metadata Document Record and/or Metadata Document (action K)

3. Naming convention

As mentioned before, the MetaRepo currently only offers basic Data Provenance support via versioning. This means, changes of a single Metadata Record are tracked, but interlinking Metadata Records to build up a more complex Data Provenance chain is only planned for a future release. The Deliverables D16.3 and D16.6, due at Month 18 and 48 respectively, will describe the good practices for Data Provenance and the Data Provenance Tools for the NEP community.

To be able to reflect a Data Provenance chain in the current version of the MetaRepo, we decided to apply a naming convention to Metadata Record IDs. The pattern is built in an incremental way, appending the Measurement (m) and the Analysis (a) to the Proposal ID (p), each of them including the version of the Metadata Document and the Metadata Schema used. In case of Metadata Documents describing Datasets generated from Joint Activities, not related to any Proposal, the Proposal ID can be substituted by the Joint Activity (ja) number. The following incremental example, summarised in Table 1, shows how the naming convention is applied to the identifiers for the records.

Let’s suppose the Research User is a member of the Proposal with ID 42. The Proposal Metadata Schema has been already registered at the MetaRepo in a Record with ID “nep_proposal” and with

relatedResource property of the Record pointing to the Proposal page on the NFFA-Europe website [11]. The Research User registers at the MetaRepo the Proposal Metadata Document automatically mapped from the database dump in a Record with Metadata Schema ID "nep_proposal" and with unique record ID "p42_1_nep_proposal", where "p" stands for proposal "42" is the proposal ID, "_1" indicates that this is version 1 of the document, and "nep_proposal" indicates the ID of the used Metadata Schema.

The reason for including a version number in the naming pattern is caused by the way the MetaRepo realises versioning. As soon as the content of a Metadata Document is changed, the version of the corresponding Metadata Record is incremented. Accessing a Metadata Record returns the most recent version by default, unless the Research User provides a version number as a query parameter. If we now use the built-in versioning of the MetaRepo, "p42" may change over time. This may cause issues in the Provenance chain as the result of obtaining "p42" can change over time as no one can know on which version a certain follow-up step has been performed. This might not be that critical for Proposal Metadata, but gets more relevant later on for Dataset metadata and analysis results. That's why we decided to include the version in our naming convention until proper interlinking of Metadata Records including their version information is possible. For now, if the metadata of "p42_1" changes, a new Metadata Record with id "p42_2" is created, while everything based on "p42_1" remains unaffected.

It is worth reminding that, as a data resource can be described according to different Metadata Schemas: depending on the purpose, different metadata can be relevant. Thus, there can be multiple Metadata Document Records related to the same resource. To have an appropriate Data Provenance, it is relevant to keep track of it in the naming convention of each Metadata Document Record ID.

For example, the Research User now generates some Raw Data: a Dataset using Magnetic Resonance Imaging (MRI), and stores it on Zenodo at a certain URL. A Metadata Schema has been created with features extracted from the DICOM file [7] which are useful for future Machine Learning applications. This Metadata Schema has been already registered at the MetaRepo with Metadata Schema Record ID "mri_ml". The Research User registers the Metadata Document describing the MRI Dataset at the MetaRepo in a Record with schema ID "mri_ml". As the Dataset is associated with Proposal 42, the unique record ID for the Metadata Document is "p42_1_nep_proposal_m1_1_mri_ml", where "p42_1_nep_proposal" stands for the proposal ID 42 version 1 based on "nep_proposal" Schema, "m1" stands for measurement number 1, "_1" indicates that this is version 1 of the document, and "mri_ml" indicates the ID of the used Metadata Schema. The relatedResource property of the Record may point to the dataset URL on Zenodo or any equivalent Data Repository. It's important to mention that the measurement number is assigned just in a sequential order at registration time, without any relation to the Dataset name or the sequence during an Experiment.

In addition, for future publication purposes, it can be useful to have Datasets described by the DataCite Metadata Schema, which has been already registered at the MetaRepo with ID "datacite". Thus, the Research User uploads a DataCite Metadata Document with schema ID "datacite" for the new Metadata Record with ID "p42_1_nep_proposal_m1_1_datacite". Now the Research User finds three MetadataRecords in the MetaRepo: one with ID "p42_1_nep_proposal" containing the Proposal Metadata Document following the "nep_proposal" Metadata Schema, one with ID "p42_1_nep_proposal_m1_1_mri_ml" containing the MRI Dataset Metadata Document following the "mri_ml" Metadata Schema, and one with ID "p42_1_nep_proposal_m1_1_datacite" containing the MRI Dataset Metadata Document following the "datacite" Metadata Schema.

The Research User performs a Data Analysis on the MRI Dataset and obtains as output some Analysed Data. The Metadata Schema for the Data Analysis has been already registered at the MetaRepo in a Record with ID "analysis". The Research User registers the Analysis Metadata Document (following the "analysis" Metadata Schema) at the MetaRepo in a Record with schema ID "analysis" for the new Metadata Document Record with ID "p42_1_nep_proposal_m1_1_mri_ml_a1_1_analysis", thus appending "a1_1_analysis" to the parent resource "p42_1_nep_proposal_m1_1_mri_ml", to indicate that this document (in version 1) describes the analysis 1 (following the "analysis" schema) performed on the Dataset generated from measurement 1 (in version 1, following the "mri_ml" schema) related to the proposal with ID 42 (in version 1, following the "nep_proposal" schema).

The Research User may perform a second (different) Data Analysis. If the analysis is performed on the MRI Dataset, the parent Record is "p42_1_nep_proposal_m1_1_mri_ml", thus the unique record ID of the new Record is "p42_1_nep_proposal_m1_1_mri_ml_a2_1_analysis", where the last analysis has sequence number "2" on the same Dataset (in the sequential order at registration time). If instead the analysis is performed on the Analysed Data, output of the first analysis performed on the MRI Dataset, the parent resource is "p42_1_nep_proposal_m1_1_mri_ml_a1_1_analysis" and the unique record ID of the new resource is "p42_1_nep_proposal_m1_1_mri_ml_a1_1_analysis_a1_1_analysis", where the last "a1_1_analysis" is appended to describe the last analysis. It is important to remark that the sequential numbering is related to the previous element of the same level. In this case, the last analysis has sequence number "1" because it's the first one performed on the Analysed Data.

Table 1: Summary of the naming convention adopted for the ID of the Metadata Document Records in the MetaRepo. The first column indicates the Metadata Document, the second column indicates the corresponding name of the Record ID, and the third column indicates the ID of the related Metadata Schema Record

Metadata Document	Metadata Document Record ID	Related Metadata Schema ID
NEP Proposal with ID 42	p42_1_nep_proposal	nep_proposal
NEP Joint Activity 6	ja6_1_nep_ja	nep_ja
Raw Data for Machine Learning applied to MRI	p42_1_nep_proposal_m1_1_mri_ml	mri_ml
Raw Data following DataCite for publication	p42_1_nep_proposal_m1_1_datacite	datacite
Analysed Data after Analysis on Raw Data	p42_1_nep_proposal_m1_1_mri_ml_a1_1_analysis	analysis
Analysed Data after (different) Analysis on Raw Data	p42_1_nep_proposal_m1_1_mri_ml_a2_1_analysis	analysis
Analysed Data after Analysis on (previous) Analysed Data	p42_1_nep_proposal_m1_1_mri_ml_a1_1_analysis_a1_1_analysis	analysis

4. MetaRepo Authorization and Authentication

4.1 Single Sign On (SSO)

As any other authenticated service, access to the MetaRepo will be mediated by the NEP SSO system based on the open-source package Keycloak [8].

To make the Keycloak SSO available for the MetaRepo, additional configuration and integration has been applied. At first, a Keycloak Client was configured with the goal to use it later on to issue authentication tokens for operations on the MetaRepo. The Keycloak Client uses the openid-connect protocol for authentication handshakes. As these handshakes involve user interactions (i.e., adding username and password if required) and redirects (i.e, redirect to the Keycloak login page and back to the requesting service) which are preferably handled by a Web Browser, the Keycloak Client is focussed on being used by the MetaRepo GUI. As the MetaRepo GUI is implemented in JavaScript, the Keycloak Client had to be configured for being public, as providing a client-specific secret restricting the access to this particular client does not make any sense as it would have to be hardcoded in the Javascript, which is readable by any user.

Since every Metadata Document is linked to a NEP Proposal, the Proposal management service adds the list of Proposal IDs to the user token in the form of "User Attribute", that is then passed in the JWT token. In such a way, the MetaRepo is aware of the user authorization level, in particular regarding editing rights.

After configuring the Keycloak Client, integration into the MetaRepo GUI took place. We used the official Keycloak JavaScript adapter parameterized to connect to the NEP Keycloak instance using the previously created Keycloak Client for authentication.

4.2 Access to Resources

Access to resources, i.e., Metadata Records in the MetaRepo, is defined in access control lists which are part of a resource's metadata. These access control lists are a simple mapping between a unique identifier, e.g., a user or group identifier, and a permission enumeration, i.e, ADMINISTERATE, WRITE, READ and NONE. The permission defines which operations can be performed on a particular resource, where ADMINISTERATE grants full access, WRITE allows to read and modify a resource, and READ only allows to read a resource.

In the MetaRepo, initial access control information is added at creation time of a resource. Thereby the creator of the record receives ADMINISTERATE permissions, whereas the proposal group receives WRITE permissions. Assigning group permissions at this point has the advantage that administration effort is reduced on the level of the MetaRepo as proposal memberships are centrally managed at Keycloak [8]. However, if required, the option of assigning fine-grained permissions on the user-level still exists. This allows adding user-specific permissions overwriting permissions a Research User obtains by proposal membership.

To allow the MetaRepo to decide on access permissions, a Research User has to log in using the Single Sign-On service of NEP (described in Section 4.1). This happens automatically as soon as the Research User tries to access the MetaRepo GUI (Section 2). The Research User is redirected to the Keycloak login page, where username and password must be provided. If authentication succeeds, the Research User is redirected back to the MetaRepo GUI which also receives a JSON

Web Token (JWT) [9] that is used for further authentication towards the MetaRepo. This JWT contains a number of so-called claims, which are basically attributes in a standardised JSON object. Among these attributes, the unique user ID and the list of proposals the user is a member of are used by the MetaRepo to decide whether a Research User is authorised to obtain a certain resource.

For security reasons, JWTs have a very short lifetime (a few minutes) which helps to limit the time for potential misuse in case a JWT has been compromised. Thus, JWTs have to be renewed frequently, which happens automatically without the need for user interaction.

Summarising, Keycloak offers an elegant way to keep user and group management at one trusted place and allows only to consume this information by VA services. Thus, any changes to Research Users, e.g., proposal memberships, can be organised centrally and are immediately propagated to consuming services at their next call.

5. MetaRepo Access Monitoring

MetaRepo is offered as a VA service to authenticated users, and its usage is monitored by counting the Units of Access to the service. A Unit of Access for the MetaRepo is defined as the single call or action (e.g. upload, query, retrieve) performed on the service by an authenticated user.

In order to distinguish authenticated from non-authenticated access, public resources will be indexed in a higher-level search engine at a later stage. In this way, direct access to the MetaRepo (for authenticated users) will be monitored and calls to the VA service will be aggregated, while the higher-level access via the search index (for non-authenticated users) will not be monitored.

MetaRepo uses a filter to detect each call to the service. Each access detected by the filter is logged by the NEP backend, whereby as minimal private data as possible is stored. Currently, only the ID of the authenticated user and the ID of the VA service are transmitted. The NEP backend provides a REST endpoint to collect this data from an authenticated user, and additional data can also be provided.

As pointed out in the NEP privacy policy [10], the MetaRepo collects monitoring data only in order to send them to the NEP backend, without storing them locally. As described in Section 4.2, the MetaRepo stores access control lists for every Metadata Document, which contain the proposal ID and the user ID in order to grant/deny access to a specific resource. Any sensitive data contained in Metadata Documents stored in the MetaRepo are neither checked or analysed, and are under the responsibility of the creator of the resource.

REFERENCES

[1] MetaStore documentation:

<https://github.com/kit-data-manager/metastore2/blob/metastoreGui/restDocu.md>

[2] DataCite Metadata Schema: <https://schema.datacite.org>

- [3] NEP Deliverable D1.1: Data Management Plan (DMP), downloadable at <https://www.nffa.eu/outcomes/deliverables/>
- [4] NEP MetaRepo: <https://metarepo.nffa.eu>
- [5] NFFA-Europe portal: www.nffa.eu
- [6] Metadata Editor: <https://github.com/kit-data-manager/metadata-editor>
- [7] DICOM <https://www.dicomstandard.org>
- [8] Keycloak: <https://www.keycloak.org>
- [9] JSON Web Token: <https://jwt.io>
- [10] NEP privacy policy: <https://nffa.eu/apply/privacy/>
- [11] NEP Proposal page: <https://nffa.eu/ar/public/proposals>

