D5.3

First version of E-CORRIDOR platform and test bed

WP5 – E-CORRIDOR Platform: Requirements / Architecture / Implementation and integration

E-CORRIDOR

Edge enabled Privacy and Security Platform for Multi Modal Transport

Due date of deliverable: 31/05/2022
Actual submission date: 31/05/2022

31/05/2022

Version 3.0

Responsible partner: HPE
Editor: Mirko Manea
E-mail address: mirko.manea at hpe.com

Project co-funded by the European Commission within the Horizon 2020 Framework Programme

	Dissemination Level	
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Authors: Claudio Caimi, Mirko Manea, Patrizia Ciampoli (HPE), Paolo Mori, Ilaria Matteucci (CNR), Thanh Hai Nguyen, Jean-Paul Bultel (CEA), Christian Plappert (FhG), Stefano Sebastio (UTRC)

Approved by: Paolo Mori, Fabio Martinelli (CNR), Florian Fenzl, Christian Plappert, Roland Rieke (FhG)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	09-Mar-2022	M. Manea, P. Ciampoli, C. Caimi	HPE	Initial ToC
0.2	09-May-2022	P. Mori, S. Sebastio,	CNR, UTRC	Merged contribution
0.3	13-May-2022	T.H. Nguyen, J.P. Bultel	CEA	Merged contribution
2.0	18-May-2022	M. Manea	HPE	Available for internal review
3.0	31-May-2022	M. Manea, F. Martinelli	HPE, CNR	Released

Executive Summary

The purpose of this document is to describe the release of the E-CORRIDOR Framework software artefacts for deliverable D5.3.

First we recall the high-level architecture discussed in D5.4, with the five E-CORRIDOR subsystems: Information Sharing Infrastructure - ISI, Information Analytics Infrastructure - IAI, Data Sharing Agreement (DSA) Lifecycle Infrastructure - DLI, Common Security Infrastructure - CSI, and Advanced Security Infrastructure - ASI. For each subsystem, in Section 4 we show the delivered software artefacts along with instructions about how to use them.

We also discuss the pre-requisites needed for proper installation of the E-CORRIDOR Framework in Section 3.

Currently at M24, the major functionalities of E-CORRIDOR are working properly, which include:

- ISI: ability to submit data, attach DSA policies to that data, and provide policy enforcement capabilities, like possibility to GET data obeying to the constraints defined by the DSA;
- IAI: ability to invoke data analytics on shared data provided by ISI and to store the analytics result on the ISI for later retrieval (subject to DSA policies). Available analytics are reported in D7.2 and includes the components packages as micro-services via the Analytics Toolbox (see 4.2);
- DLI: ability to create DSA policies via the DSA Editor and provide them to the ISI for data sharing and enforcement;
- CSI: ability to provide the Identity Manager services, including the authentication layer based on OpenID Connect and Keycloak, as well as the Key and Encryption Manager and the off-the-shelf Secure Auditing tool;
- ASI: ability to provide an initial version of the Advanced Security services related to Privacy-Aware Seamless Multimodal Authentication, Continuous Behavioural Authentication, and Privacy-Aware Interest-Based Service Sharing.

Table of contents

1. Introduction

This document presents the content of Deliverable D5.3, which is the actual software artefacts released for the E-CORRIDOR Framework at its first version of M24. The components are those designed in the previous deliverables D5.2 and D5.4, as well as those coming from WP6 and WP8.

We include the installation procedures about how to deploy the released Framework and the operational procedures that can be followed for performing some tests of the correct Framework installation.

1.1. Deliverable Structure

The document is structured as follows:

- Section 1 is this introduction;
- Section 2 reports the high-level architecture presented in D5.4;
- Section 3 describes the delivered software artefacts and the pre-requisites needed for their deployment;
- Section 4 shows the installation guidelines for the Framework;
- Section 5 illustrates some operational procedure that can be used to test the basic features of the Framework;
- Section 6 draws the conclusions and Section 7 reports the bibliography.

1.2. Definitions and Abbreviations

Term	Meaning
AES	Advanced Encryption Standard
ADS-B	Automatic Dependent Surveillance-Broadcast
AT	Airport and integrated Train transport (WP2 pilot)
BCBP	Bar-Coded Boarding Pass
C3ISP	Collaborative and Confidential Information Sharing and Analysis for Cyber Protection
CAN	Controlled Area Network
CAPEC	Common Attack Pattern Enumeration and Classification
CEF	Common Event Format
COTS	Commercial Off-The-Shelf
CPE	Common Platform Enumerations
CTI	Cyber Threat Information
DoA	Description of Action
DMO	Data Manipulation Operations
DSA	Data Sharing Agreement
DUCS	Data Usage Control System
EML	Electronic Mail

eIDAS	electronic Identification, Authentication and trust Services
EU	European Union
FMC	Fundamental Modelling concepts
FHE	Fully Homomorphic Encryption
GDPR	General Data Protection Regulation
GPS	Global Positioning System
GPX	GPS Exchange Format
HDFS	Hadoop Distributed File System
HTTP	Hyper-Text Transfer Protocol
IoT	Internet of Thing
IAI	Information Analytics Infrastructure
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
IIT	*Istituto di Informatica e Telematica* at CNR
ISAC	Information Sharing and Analytics Centre (WP4 pilot)
ISI	Information Sharing Infrastructure
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JPEG	Joint Photographic Experts Group
KMS	Key Management System
LAS	LASer format
M2M	Machine to Machine
MoSCoW	“Must have”, “Should have”, “Could have”, “Won’t have but would like”
MMT	Multi-Modal Transport
MRH	Multi Resources Handler
NMEA	National Marine Electronics Association
NFR	Non-Functional Requirement
ODB	On Board Diagnostics
OWASP	Open Web Application Security Project
OWL	Ontology Web Language
PDP	Policy Decision Point
REST	REpresentational State Transfer
RFID	Radio-frequency identification
RPC	Remote Procedure Call

RSSI	Received Signal Strength
SAML	Security Assertion Markup Language
S2C	Car Sharing in Smart Cities (WP3 pilot)
SSO	Single Sign On
STIX	Structured Threat Information eXpression
SUCS	Service Usage Control System
TPM	Trusted Platform Module
UC	Use Case
US	User Story
UUID	Universally Unique Identifier
VDL	Virtual Data Lake
VM	Virtual Machine
XACML	eXtensible Access Control Markup Language

2. High-Level Architecture

The purpose of showing the architecture in this kind of deliverable (which should present the first delivered software artefacts) is to recall that the E-CORRIDOR framework is made of different subsystems, each of them has a set of software services (modules) implemented. For this reason, Section 4 splits the installation procedure per subsystem. Although the subsystems have reached different maturity levels, we strive to uniform the procedures as much as possible, and to provide a fully homogenous installation steps by the last version of E-CORRIDOR, i.e. deliverable D5.5.

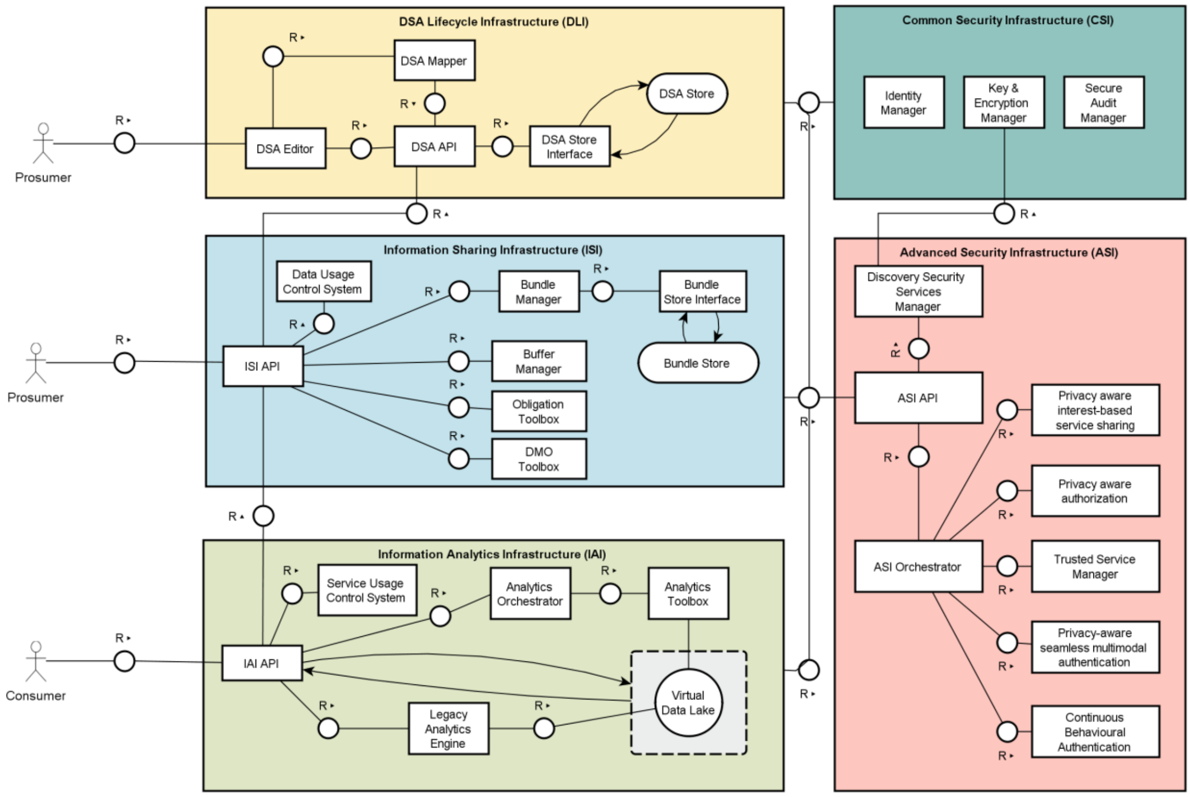

DSA Lifecycle Infrastructure (DLI)
DSA Mapper
DSA Editor
DSA API
DSA Store
DSA Store Interface
Prosumer
Common Security Infrastructure (CSI)
Identity Manager
Key & Encryption Manager
Secure Audit Manager
Information Sharing Infrastructure (ISI)
Data Usage Control System
ISI API
Bundle Manager
Bundle Store Interface
Bundle Store
Buffer Manager
Obligation Toolbox
DMO Toolbox
Prosumer
Advanced Security Infrastructure (ASI)
Discovery Security Services Manager
ASI API
ASI Orchestrator
Privacy aware interest-based service sharing
Privacy aware authorization
Trusted Service Manager
Privacy-aware seamless multimodal authentication
Continuous Behavioural Authentication
Information Analytics Infrastructure (IAI)
Service Usage Control System
Analytics Orchestrator
Analytics Toolbox
IAI API
Virtual Data Lake
Legacy Analytics Engine
Consumer
R

Figure 1: E-CORRIDOR high-level architecture - version 2 (Month 24)

The architecture in Figure 1 allows us to also illustrate the entry points to each subsystem (e.g., the API boxes) that are used in Section 5 to provide the operational procedures that can be used to test and integrated with the E-CORRIDOR framework.

3. Delivered E-CORRIDOR Software

This section shows the pre-requisites needed for the correct deployment of the E-CORRIDOR framework as released at M24. It also provides a glance about how the source code is compiled and packaged as a micro-service into the multi-container application pattern used in E-CORRIDOR (see D5.4, Section 3.1).

3.1. Pre-requisites

The E-CORRIDOR software components and modules are packaged as micro-services running on containers. As a pre-requisites for E-CORRIDOR installation, the following must be available:

- One or mode Linux machines, either physical or virtual:
    - We tested Ubuntu Linux Server 20.04 LTS 64-bits running on Virtual Machines:
        - https://releases.ubuntu.com/20.04.4/ubuntu-20.04.4-live-server-amd64.iso
- From a basic installation of Linux (Ubuntu), the following tools are needed:
    - Docker Engine Community Edition:
        - https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository
    - Docker Compose:
        - https://docs.docker.com/compose/install/#install-compose

The (virtual) hardware characteristics of the machine are quite modest in order to have a testing environment where all the services start up. The following is the bare minimum to have an all-in-one E-CORRIDOR deployment, since high-computation and memory requirements are needed to proficiently use some of the E-CORRIDOR services, like the data analytics or the advanced encryption capabilities.

Table 1 - Hardware characteristics

Hardware	Requirement
CPU	4
RAM	8 GB
Storage	200 GB

3.2. Software Artefacts

The current source code of the E-CORRIDOR framework is kept in a private versioning system running on a GitLab instance hosted at CNR partner. As reported in Section 10.1 of D5.4, the GitLab defines a tree-like structure where the source code is hosted. Most of the source code is written in Java using Spring Boot [2] as a micro-service framework and Maven [3] as a build tool. This helps a lot, because the compile, test and package phases are always the same with a simple command line:

```
$ mvn clean package
```

This generates the executable Java file (typically a .jar file). The executable Java file is finally packages into a Docker container and published to a Docker Registry (in our case we use Nexus, as reported in Section 10.1.1 of D5.4. Also this step is simple, once the Docker manifest file has been prepared, e.g.:

```
$ docker build -f docker/Dockerfile -t
devecorridor.iit.cnr.it:5000/ecorridor/dli/dsa-store-api:latest .
$ docker push devecorridor.iit.cnr.it:5000/ecorridor/dli/dsa-store-
api:latest
```

However, as we previously said, an E-CORRIDOR subsystem is an orchestration of several pieces (micro-services) that are made as containers. In fact, each subsystem takes part of a Docker Compose multi-container application that manages (starts-up, shutdown, etc.) all the micro-services. So a Docker Compose manifest file is required for each micro-service which is assembled together when they are run, like in the following example:

```
$ docker-compose -p s2c-DLI --env-file env-s2c -f docker-compose_dsa-api.yml
… -f docker-compose_dsa-vocabulary.yml up -d
```

Section 3 of D5.2 describe the Docker Compose approach in more details.

Nevertheless, all this process is automated for the E-CORRIDOR partners, because we use Jenkins to perform these actions, where the package, build and push phases happens under the scenes when a developer decides to release a source code change. The Jenkins pipelines are described in Section 10.1.2 of D5.2. Jenkins also allows to deploy, i.e., install the built software to a target environment.

3.3. Download E-CORRIDOR

We prepared a website page where the current version of E-CORRIDOR artefacts can be downloaded:

- https://devecorridor.iit.cnr.it/releases/e-corridor_M24.tar.gz

The page is password-protected and credentials will be communicated to interesting parties.

4. Installation procedures

In the environments used for Piloting (AT/S2C/ISAC VMs), we have automated the installation procedures (deployment) via the CI/CD pipelines implemented with Jenkins (see D5.4, Section 10.1.2). However, it is still possible to install manually for testing the framework, e.g. in a dedicated VM satisfying the requirements reported in Section 3.1. We released the binary artefacts that can be packaged in a container, since releasing the full container image requires usually a significant amount of space.

4.1. ISI - Information Sharing Infrastructure

The next table lists the artefacts released for ISI:

Table 2 - ISI artefacts

Component	Module	Artefact
ISI API	-	isi-api.war
Data Usage Control System	Event Handler	event-handler.war
Data Usage Control System	MRH	multi-resource-handler.war
Data Usage Control System	DSA Adapter	dsa-adapter-frontend.war
Bundle Manager	-	bundle-manager.war
Bundle Store Interface	-	dpos-api.war
Buffer Manager	-	buffer-manager.war

They can be packaged into a container, but they can also be used as standalone web application archives (WAR) when deployed into a Servlet engine (e.g., Apache Tomcat). We plan to package appropriately all of them as containers in the next period, since that for now we focused on making the functionalities working.

4.2. IAI - Information Analytics Infrastructure

The next table lists the artefacts released for IAI:

Table 3 - IAI artefacts

Component	Module	Artefact
IAI API	-	iai-api.war
Analytics Toolbox	Itinerary planner	witp.jar
Analytics Toolbox	CO2 calculator	CO2calculator/app/index.js
Analytics Toolbox	Driver DNA	server.py
Analytics Toolbox	CFA	cfa (Linux binary)
Analytics Toolbox	Face recognition	server.py, iai_test_client.py
Analytics Toolbox	AutomotiveIDS	server.py, AutomotiveIDS.py, iai_test_client.py

Analytics Toolbox	Driving license checker	dap-analysis-pip.jar

Note: the analytics modules of the Analytics Toolbox follow the plugin architecture described in Section 5.4 of D6.2. Further instructions are bundled with the released archive.

4.3. DLI - DSA Lifecycle Infrastructure

The next table lists the artefacts released for DLI:

Table 4 - DLI artefacts

Component	Module	Artefact
DSA API	-	dsa-api.jar
DSA Editor	DSA Editor	dsa-editor.jar
DSA Editor	DSA Editor Frontend	dsa-editor-frontend.war
DSA Editor	Vocabularies	dsa-vocabulary.jar
DSA Mapper	-	dsa-mapper.war
DSA Store Interface	-	dsa-store-api.jar

In addition, the DLI uses the following out-of-the-box components: a MariaDB database and a MongoDB database.

All the DLI components are packaged as containers and orchestrated by Docker Compose.

4.4. CSI - Common Security Infrastructure

The next table lists the artefacts released for CSI:

Table 5 - CSI artefacts

Component	Module	Artefact
Identity Manager	Keycloak	export-ecorridor-realm.json
Key and Encryption Manager	CSI Core API	ke-core-manager-api.war
Key and Encryption Manager	DPOS KEY API	dpos-key.war
Key and Encryption Manager	DPOS Encryption API	dpos-encryption.war

In particular, for Keycloak we provide the configuration that can be imported in an empty instance to make working the Operational Procedures explained in Section 5.

4.5. ASI - Advanced Security

The next table lists the artefacts released for ASI:

Table 6 - ASI artefacts

Component	Module	Artefact
Continuous Behavioural Authentication	Node, connector, proxy, identity provider, service provider	eidasNode.war, IdP.war, SP.war, connector.war, proxy.war
Privacy aware seamless multimodal authentication	Engine[1]	engine (binary)
Discovery Security Service Manager	Discovery services & multi instances management	bigpi-discovery-service.jar
ASI Orchestrator	Gateway ASI API & ASI Orchestrator	bigpi-gateway-service.jar
Privacy-aware Interest-based service sharing	Similarity Matching	Binary java

[1] This component foresees the presence of the "reasoner" module not yet available

5. Operational Procedures

This section describes how the installed E-CORRIDOR Framework can be used/tested and how it can be integrated into/used by an application. To support these needs, E-CORRIDOR provides **external APIs** based on the RESTful web services paradigm, where there is a list of http endpoints that can be called programmatically. To easier testing, developers/testers can call the APIs through a simple web interface that describes their signatures and allows specifying the required parameters.

Since APIs are protected by the Identity Manager component (in particular by Keycloak), it is necessary to get an access token when issuing the RESTful calls. This can be achieved easily with the following snippet using the CURL [4] shell command:

```
curl   -d   'client_id=api-services'   -d   'client_secret=a0418143-94a8-46e8-ab3d-
d56d01b7a028'    -d    'username=my_username'    -d    'password=my_password    -d
'grant_type=password'    -d    'scope=openid    profile    ecorridor    email'
'https://ecorridor.iit.cnr.it/auth/realms/ecorridor/protocol/openid-connect/token'
```

This returns the access token in this format:

```
   "access_token": "eyJhbGciOiJS…..."
```

Note: in the command above, the URL ecorridor.iit.cnr.it is used, but this depends on the host where the CSI components have been installed (in particular Keycloak).

5.1. *ISI API*

This section describes the invocation of the main API methods of the ISI. For each of these methods, in the following we show the URL to be invoked, the parameter to be passed, an invocation of the method using the CURL [4] shell command with realistic parameters values, and the expected response returned by the method. A more detailed description of the methods and of the parameters can be found in deliverable D6.2.

Note: in the examples below, the URL ecorridor.iit.cnr.it is used, but this depends on the host where the ISI API has been installed.

Create Data Bundle

URL: https://ecorridor.iit.cnr.it/isi-api/v1/dpo

METHOD: **POST**

PARAMETERS:

- **fileToSubmit**: path of the file that contains the data to be embedded in the Data Bundle that is being created;
- **input_metadata**: JSON formatted string embedding a set of metadata.

CURL COMMAND TO TEST THE OPERATION:

```
curl --location --request POST 'https://ecorridor.iit.cnr.it/isi-api/v1/dpo/' \

--header 'Authorization: Bearer [token-value]' \

--form
'input_metadata="{\"Request\":{\"Attribute\":[{\"AttributeId\":\"urn:oasis:names:tc
:xacml:1.0:resource:data-start-date\",\"Value\":\"2022-05-
30\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:names:tc:xacml:1.0:reso
urce:data-start-
time\",\"Value\":\"10:30:00\"\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis
:names:tc:xacml:1.0:resource:data-end-date\",\"Value\":\"2022-05-
30\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:names:tc:xacml:1.0:reso
urce:data-end-
time\",\"Value\":\"12:30:00\"\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis
:names:tc:xacml:3.0:resource:dsa-id\",\"Value\":\"DSA-1f6eb2c4-8e18-418f-a7f5-
8849179cd119\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:names:tc:xacm
l:3.0:resource:resource-
type\",\"Value\":\"\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:names:
tc:xacml:3.0:resource:resource-
owner\",\"Value\":\"CNR\",\"DataType\":\"string\",},{\"AttributeId\":\"file:extensi
on\",\"Value\":\"json\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:name
s:tc:xacml:1.0:subject:subject-
id\",\"Value\":\"user\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:name
s:tc:xacml:1.0:action:action-
id\",\"Value\":\"create\",\"DataType\":\"string\",},{\"AttributeId\":\"urn:oasis:na
mes:tc:xacml:3.0:subject:access-
purpose\",\"Value\":\"generic\",\"DataType\":\"string\",},]}}"' \
--form 'fileToSubmit=@"/tmp/test_upload"'
```

Note: the ***[token-value]*** shall be replaced by the token generated by the Identity Manager component (in particular by Keycloak).

EXPECTED RESPONSE:

When the creation is successfully performed the returned response code is 201, and the payload contains the following JSON string:

```
{
  "status": "SUCCESS",
  "content": {
    "additionalProperties": {
      "dpoId": "1653116174461-dd603f0f-d9d6-40fd-9cf9-a11a789375d9"
    }
  }
}
```

The JSON string contains the status of the request, which is SUCCESS in this case, and the ID assigned to the Data Bundle that has been created (the value of the dpoId field).

Read Data Bundle

URL: https://ecorridor.iit.cnr.it/isi-api/v1/dpo/1652978489181-fb39371f-0eff-4ceb-a50a-2d49e52a88ef/

METHOD: **GET**

PARAMETERS:

- **X-ecorridor-input_metadata**: JSON formatted string embedding a set of metadata. It is passed in the Header:
- **Dpo-id**: ID of the data bundle to be read. It is passed as suffix of the URL

CURL COMMAND TO TEST THE OPERATION:

```
curl      --location      --request      GET      'https://ecorridor.iit.cnr.it/isi-
api/v1/dpo/1652978489181-fb39371f-0eff-4ceb-a50a-2d49e52a88ef/' \

--header  'X-ecorridor-input_metadata:  {  \"Request\":  {  \"Attribute\":  [  {
\"AttributeId\":  \"urn:oasis:names:tc:xacml:1.0:subject:subject-id\",  \"Value\":
\"giuseppe.crincoli\",  \"DataType\":  \"string\"      },{  \"AttributeId\":
\"urn:oasis:names:tc:xacml:1.0:action:action-id\",      \"Value\":      \"read\",
\"DataType\":          \"string\"          },          {          \"AttributeId\":
\"urn:oasis:names:tc:xacml:3.0:subject:access-purpose\",  \"Value\":  \"generic\",
\"DataType\":                              \"string\"                            },
{\"AttributeId\":\"urn:oasis:names:tc:xacml:3.0:subject:subject-
organisation\",\"Value\":\"CNR\",\"DataType\":\"string\"} ] } }' \

--header 'Authorization: Bearer [token-value]'
```

Note: the ***[token-value]*** shall be replaced by the token generated by the Identity Manager component (in particular by Keycloak).

RESPONSE:

When the read operation is successfully performed the returned response code is 200, and the payload contains the Data Bundle content.

5.2. IAI API

This section describes the invocation of the main API methods of the IAI. For each of these methods, in the following we show the URL to be invoked, the parameter to be passed, an invocation of the method using the CURL [4] shell command with realistic parameters values, and the expected response returned by the method. A more detailed description of the methods and of the parameters can be found in deliverable D6.2.

Note: in the examples below, the URL ecorridor.iit.cnr.it is used, but this depends on the host where the IAI API has been installed.

Run Analytic

URL: https://ecorridor.iit.cnr.it/iai-api/v1/runAnalytics

METHOD: POST

PARAMETERS

- **param:** JSON formatted string embedding all the parameters to be passed.

Example: JSON string body that execute an analytic with name <**NAME**> and event type <**EVENT_TYPE**>:

```
{
    "additionalAttribute":{
       "accessPurpose":"Cyber Threat Monitoring"
    },
    "metadata": {
   },
    "searchCriteria":{
       "combining_rule":"or",
       "criteria":[
          {
             "attribute":"event_type",
             "operator":"eq",
             "value":"<EVENT_TYPE>"
          }
       ]
    },
    "serviceName":"<NAME>"
}
```

CURL COMMAND TO TEST THE OPERATION:

```
curl      --location      --request      POST      'https://ecorridor.iit.cnr.it/iai-
api/v1/runAnalytics' \

--header 'Authorization: Bearer [token-value]' \

--header 'Content-Type: application/json' \

--data-raw '{

  "additionalAttribute":{

     "accessPurpose":"Cyber Threat Monitoring"

  },

  "metadata": {

 },

  "searchCriteria":{

     "combining_rule":"or",

     "criteria":[

        {

           "attribute":"event_type",

           "operator":"eq",

           "value":"carData"

        }
```

```
    ]

  },

  "serviceName":"driverdna"

}'
```

EXPECTED RESPONSE:

When the run analytics operation is successfully performed the returned response code is 200, and the payload contains the following JSON string:

```
{
    "value": "e2813d95-b429-416e-adca-4b86bd104d82",
    "message": "OK"
}
```

The string contains the ID of the session that has been created to execute the analytics. This ID will be used to query the IAI to test whether the analytics terminated and to get the results (see in the following)

Get Response

URL:

https://ecorridor.iit.cnr.it/v1/getResponse/e2813d95-b429-416e-adca-4b86bd104d82/

METHOD: **GET**

PARAMETERS:

- **Response-id**: ID of the session of which we want to get the results. It is passed as suffix of the URL

CURL COMMAND TO TEST THE OPERATION:

```
curl      --location      --request      GET      'https://ecorridor.iit.cnr.it/iai-
api/v1/getResponse/7f47c2d1-2ab4-485c-b748-18b2b60345f4/' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer [token-value]'
```

RESPONSE:

When the get result operation is successfully performed the returned response code is 200, and the payload contains the following JSON string:

```
{

  "result_bundles": [],

  "result_message": "1623,248",

  "finished": true,
  "status": "FINISH_OK"

}
```

If the value of the field “finished” is “true”, the value of the field “result_message” is a string produced by the analytics which describes the result, while the field “result_bundles” is an array embedding the IDs of the Data Bundles that have been created as results of the analytics. In this specific case, the driverdna analytics returned a textual result only.

5.3. DSA Editor

This section show the main screens of the DSA Editor web interface, which is used to define the DSA polices used to regulate the data sharing and data analytics processes. The full documentation is available in D5.4, Section 7.1 and D6.2, Section 3.1.

The DSA Editor requires authentication to access it and this is the Keycloak page of the OpenID Connect flow:

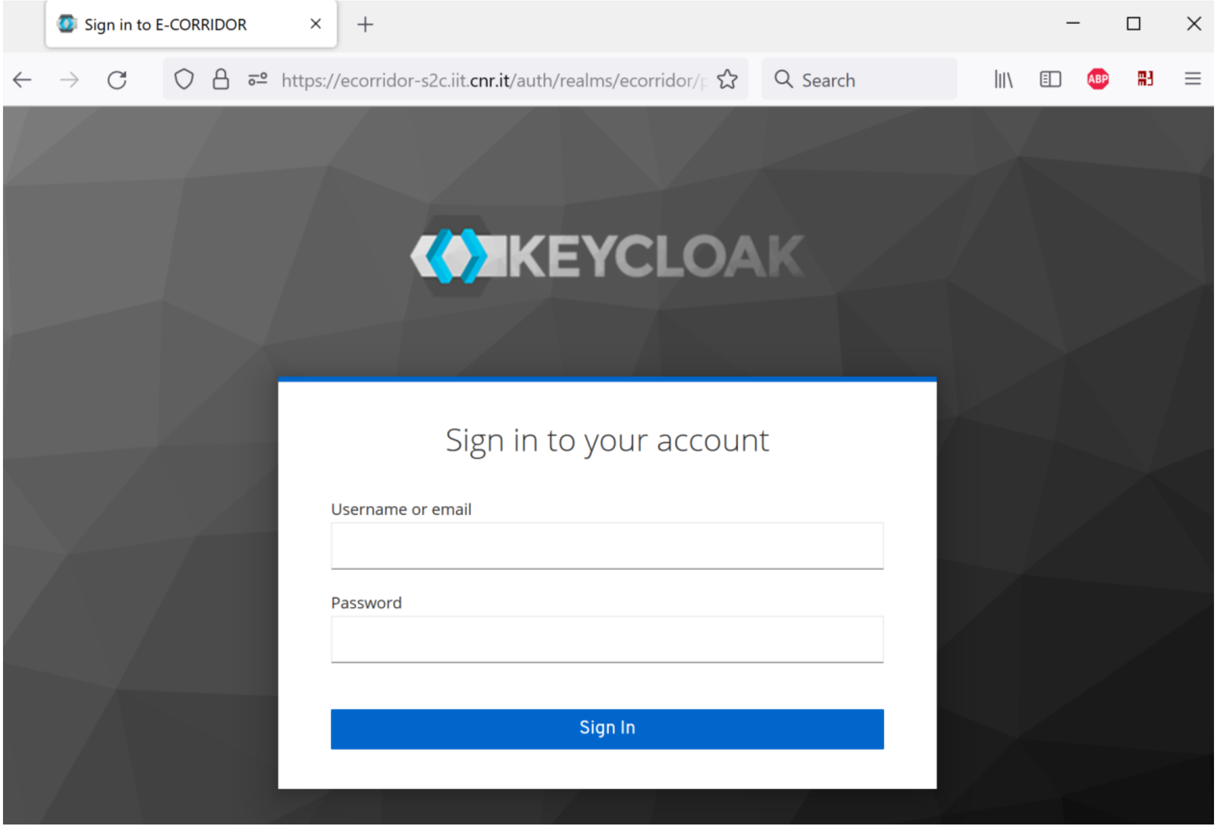
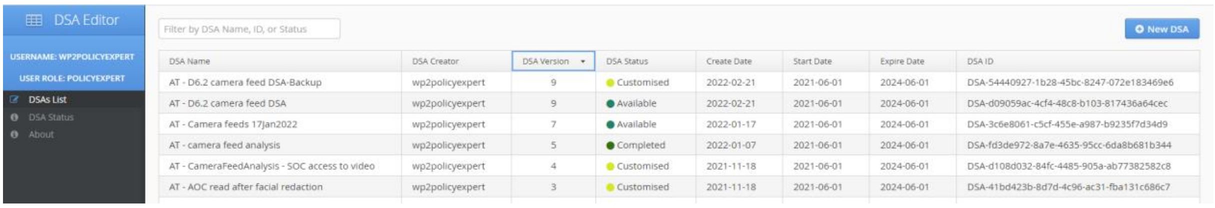


Figure 2: DSA Editor login page

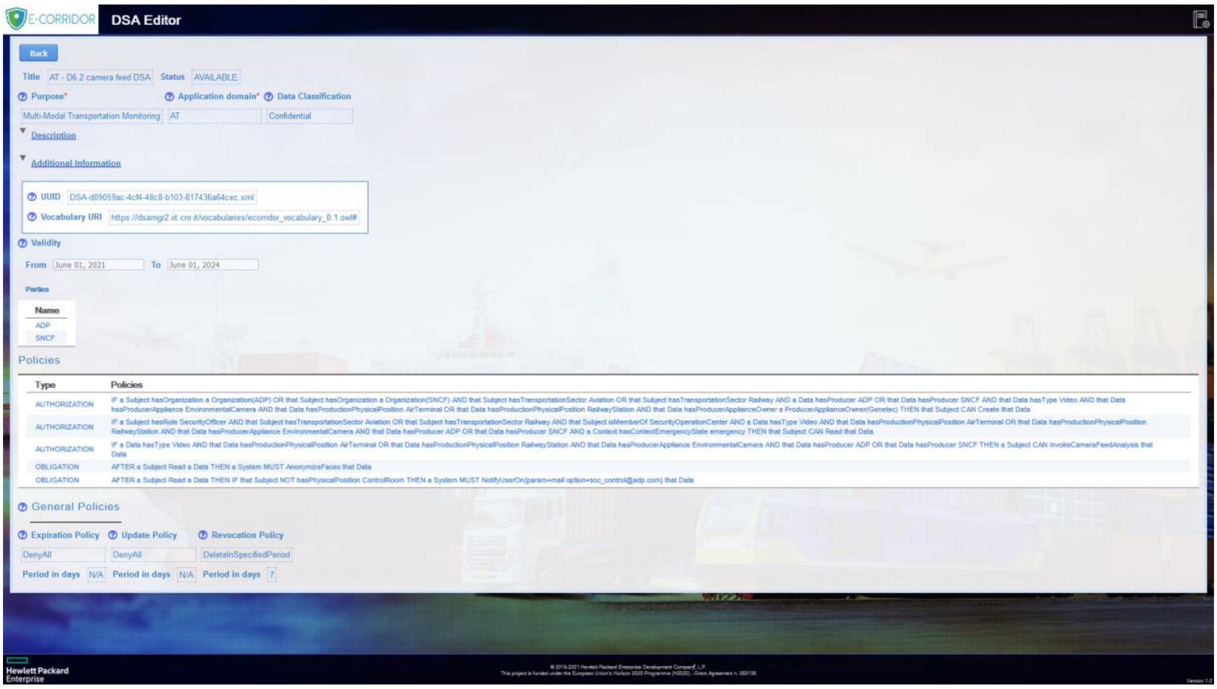
After authentication the landing page presents the list of DSA the connected user can see:



DSA Name	DSA Creator	DSA Version	DSA Status	Create Date	Start Date	Expire Date	DSA ID
AT - D6.2 camera feed DSA-Backup	wp2policyexpert	9	● Customised	2022-02-21	2021-06-01	2024-06-01	DSA-54440927-1b28-45bc-8247-072e183469e6
AT - D6.2 camera feed DSA	wp2policyexpert	9	● Available	2022-02-21	2021-06-01	2024-06-01	DSA-d09059ac-4cf4-48c8-b103-817436a64cec
AT - Camera feeds 17Jan2022	wp2policyexpert	7	● Available	2022-01-17	2021-06-01	2024-06-01	DSA-3c6e8061-c5cf-455e-a987-b9235f7d34d9
AT - camera feed analysis	wp2policyexpert	5	● Completed	2022-01-07	2021-06-01	2024-06-01	DSA-fd3de972-8a7e-4635-95cc-6da8b681b344
AT - CameraFeedAnalysis - SOC access to video	wp2policyexpert	4	● Customised	2021-11-18	2021-06-01	2024-06-01	DSA-d108d032-84fc-4485-905a-ab77382582c8
AT - AOC read after facial redaction	wp2policyexpert	3	● Customised	2021-11-18	2021-06-01	2024-06-01	DSA-41bd423b-8d7d-4c96-ac31-fba131c686c7

Figure 3: List of DSA for a sample user (wp2policyexpert)

Once a specific DSA is selected, it can be edited or shown. The following screenshot illustrates the Show of a DSA:



Title: AT - D6.2 camera feed DSA Status: AVAILABLE

Purpose*: Multi-Modal Transportation Monitoring

Application domain*: AT

Data Classification: Confidential

UUID: DSA-d09059ac-4cf4-48c8-b103-817436a64cec.xml

Vocabulary URI: https://dsamgr2.iit.cnr.it/vocabularies/ecorridor_vocabulary_0.1.owl#

Validity

From: June 01, 2021 To: June 01, 2024

Parties

Name
ADP
SNCF

Policies

Type	Policies
AUTHORIZATION	IF a Subject hasOrganization a Organization(ADP) OR that Subject hasOrganization a Organization(SNCF) AND that Subject hasTransportationSector Aviation OR that Subject hasTransportationSector Railway AND a Data hasProducer ADP OR that Data hasProducer SNCF AND that Data hasType Video AND that Data hasProducerAppliance EnvironmentalCamera AND that Data hasProductionPhysicalPosition AirTerminal OR that Data hasProductionPhysicalPosition RailwayStation AND that Data hasProducerApplianceOwner a ProducerApplianceOwner(Genetec) THEN that Subject CAN Create that Data
AUTHORIZATION	IF a Subject hasRole SecurityOfficer AND that Subject hasTransportationSector Aviation OR that Subject hasTransportationSector Railway AND that Subject isMemberOf SecurityOperationCenter AND a Data hasType Video AND that Data hasProductionPhysicalPosition AirTerminal OR that Data hasProductionPhysicalPosition RailwayStation AND that Data hasProducerAppliance EnvironmentalCamera AND that Data hasProducer ADP OR that Data hasProducer SNCF AND a Context hasContextEmergencyState emergency THEN that Subject CAN Read that Data
AUTHORIZATION	IF a Data hasType Video AND that Data hasProductionPhysicalPosition AirTerminal OR that Data hasProductionPhysicalPosition RailwayStation AND that Data hasProducerAppliance EnvironmentalCamera AND that Data hasProducer ADP OR that Data hasProducer SNCF THEN a Subject CAN InvokeCameraFeedAnalysis that Data
OBLIGATION	AFTER a Subject Read a Data THEN a System MUST AnonymizeFaces that Data
OBLIGATION	AFTER a Subject Read a Data THEN IF that Subject NOT hasPhysicalPosition ControlRoom THEN a System MUST NotifyUserOn{param=mail option=soc_control@adp.com} that Data

General Policies

Expiration Policy	Update Policy	Revocation Policy
DenyAll	DenyAll	DeleteInSpecifiedPeriod
Period in days: N/A	Period in days: N/A	Period in days: 7

Hewlett Packard Enterprise
© 2016-2021 Hewlett Packard Enterprise Development Company, L.P.
This project is funded under the European Union's Horizon 2020 Programme (H2020) - Grant Agreement n. 883135
Version 1.0

Figure 4: Show DSA page

5.4. *ASI API*

At the moment of writing this deliverable, the ASI API is not ready yet. However, this does not prevent the testing of the ASI services, since they can be called directly through their own APIs.

In fact, when available, the ASI API will be a gateway that allows to scale up a function/protocol by allowing to run independently many instances of it. It will work as a proxy service that redirects to distinct sets of methods, each set corresponding to one of the main advanced security privacy-aware services, with specific methods per service. Sections listed below are representing APIs that have been developed at component level for the moment, each one being associated with such a set of methods.

Due to this, to use the ASI API when ready, an API migration step will be necessary for the caller components and will be part of the next period activities.

5.4.1. *Privacy-Aware Seamless Multimodal Authentication*

This section describes the invocation of the main API methods of the ASI used in PASMA (Privacy-Aware Seamless Multimodal Authentication) module. Each of the examples shown below were generated with CURL [4] shell command with sample parameters.

Note: in the examples below, the URL ecorridor.iit.cnr.it is used, but this depends on the hosts where the ASI API has been installed.

Workload schema creation

URL: https://ecorridor.iit.cnr.it/asi-api/v1/orchestrator/workflow_schema/create

METHOD: **POST**

PARAMETERS:

It contains a list of entries separate for each of the commands defined in the workflow schema. Each section needs to list following parameters:

- **id**: human readable and unique workload schema identifier;
- **run**: information about running task mode (parallel or serial), its type (URL/OS) and wait time before orchestrator will automatically stop execution;
- **cmd**: JSON formatted string containing workflow step command.

CURL COMMAND TO TEST THE OPERATION:

```
curl --location --request POST 'https://ecorridor.iit.cnr.it/asi-
api/v1/orchestrator/workflow_schema/create' \
--header 'Authorization: Bearer [token-value]' \
-H "accept: application/json" \
-H "Content-Type: application/json" \
-d " [
    {\"id\": \"bgpi_analysis_result_serial\",
     \"run\": {\"mode\": \"serial\", \"type\": \"url\", \"timeout\": 300},
     \"cmd\":\"https://ecorridor.iit.cnr.it/api/dap-analysis-
result/openapi/v1/bigpi-analysis-result/<username>\"},
    {\"id\": \"bgpi_analysis_result_parallel_1\",
     \"run\": {\"mode\": \"parallel\", \"type\": \"url\", \"timeout\": 300},
     \"cmd\":\"https://ecorridor.iit.cnr.it/api/dap-analysis-
result/openapi/v1/bigpi-analysis-result/<username>\"},
    {\"id\": \"bgpi_analysis_result_parallel_2\",
     \"run\": {\"mode\": \"parallel\", \"type\": \"os\", \"timeout\": 300},
     \"cmd\":\"https://ecorridor.iit.cnr.it/api/dap-analysis-
result/openapi/v1/bigpi-analysis-result/<username>\"}
]"
```

Note: the `[token-value]` shall be replaced by the token generated by the Identity Manager component (in particular by Keycloak). The other parameter `<username>` will need to be supplied at the time when a particular instance of the workload schema is executed. This parameter will be further explained in next request. To enable more schema flexibility, it is also important to note that inside workload schema any other parameter annotated with sharp brackets can be added to depending on the given workload schema sequence requirements.

EXPECTED RESPONSE:

When the creation is successfully performed the returned response code is 201, and the payload contains the following JSON string:

```
{
  "id": "pasma_at_pilot_authentication",
  "status": "Success"
}
```

The JSON string contains the status of the request, which is **SUCCESS** in this case, and the ID assigned to the workflow schema that has been created. After successful creation the user can start running workflows with this predefined schema.

Workload execution

URL: https://ecorridor.iit.cnr.it/asi-api/v1/orchestrator/workflow /start

METHOD: **POST**

PARAMETERS:

- **id**: human readable task name;
- **schema_id**: workflow schema identifier associated with this particular task;
- **params**: parameters that workflow_schema requires to be added to work in real task execution scenario. This is a dictionary with corresponding matching keys and its values that will be automatically switched when given workflow task is executed;
- **auth**: additional service authorization fields for given task to execute.

CURL COMMAND TO TEST THE OPERATION:

```
curl      --location      --request      POST      'https://ecorridor.iit.cnr.it/asi-
api/v1/orchestrator/workflow/start' \
--header 'Authorization: Bearer [token-value]' \
-H "accept: application/json" \
-H "Content-Type: application/json" \
-d "[
      {\"id\": \"task 1\",
       \"schema_id\": \"bgpi_analysis_result_serial\",
       \"params\": {\"<username>\": \"test\"},
       \"auth\": {\"user\": \"test\", \"pass\": \"********\"}},
      {\"id\": \"task 2\",
       \"schema_id\": "bgpi_analysis_result_parallel_1",
       \"params\": {\"<username>\": \"test\"},
       \"auth\": {\"user\": \"test\", \"pass\": \"********\"}},
      {\"id\": \"task 3\",
       \"schema_id\": \"bgpi_analysis_result_parallel_2",
       \"params\": {\"<username>\": \"test\"},
       \"auth\": {\"user\": \"test\", \"pass\": \"********\"}}
]”
```

Note: the ***[token-value]*** shall be replaced by the token generated by the Identity Manager component (in particular by Keycloak). The ***<username>*** parameter will be used with the schema to execute the tasks in given serial or parallel order.

EXPECTED RESPONSE:

When the creation is successfully performed the returned response code is 201, and the payload contains the following JSON string:

```
{
  "id": "964e45ec-e53d-11ec-8fea-0242ac120002",
  "status": "Success"
}
```

The JSON string contains the status of the request and unique ID assigned to the workflow that is been executed.

Workload status check

URL: https://ecorridor.iit.cnr.it/asi-api/v1/orchestrator/workflow/status

METHOD: **GET**

PARAMETERS:

- No parameters are required.

CURL COMMAND TO TEST THE OPERATION:

```
curl --location --request GET 'https://ecorridor.iit.cnr.it/asi-api/v1/orchestrator/workflow/status' \
--header 'Authorization: Bearer [token-value]' \
```

Note: the ***[token-value]*** shall be replaced by the token generated by the Identity Manager component (in particular by Keycloak).

EXPECTED RESPONSE:

The PASMA module returns by default response code is 200, and the payload contains the following JSON string:

```
[
  {
    "id": "964e45ec-e53d-11ec-8fea-0242ac120002",
    "status": "Completed"
  },
  {
    "id": "99fd0603-8b6a-46ad-8593-b3c0ae35f522",
    "status": "Working"
  }
]
```

The JSON string contains list of the workflows that were completed (with the **Completed** status) or are currently working (with **Working** status).

5.4.2. Continuous Behavioural Authentication

This section describes CBA (Continuous Behavioural Authentication) component test procedure using SAML authentication request and CURL shell script command. The component is at the moment under development and final operational commands will be provided in latest version of this deliverable at the end of the project.

SAML Authentication request

URL: https://ecorridor.iit.cnr.it/asi-api/v1/eidas/authenticate

METHOD: **POST**

PARAMETERS:

- **data**: authentication XML file in SAML format.

Authentication request.xml file example to be used in the authentication request test:

```
<samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="pfx41d8ef22-e612-8c50-9960-
1b16f15741b3" Version="2.0" ProviderName="SP test" IssueInstant="2014-07-
16T23:52:45Z" Destination="https://ecorridor.iit.cnr.it/asi-
api/v1/eidas/authenticate"
ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
AssertionConsumerServiceURL="https://ecorridor.iit.cnr.it/asi-
api/v1/eidas/authenticate?acs">

  <saml:Issuer>https://ecorridor.iit.cnr.it/asi-api/v1/eidas</saml:Issuer>

  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

    <ds:SignedInfo>

      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>

      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

      <ds:Reference URI="#pfx41d8ef22-e612-8c50-9960-1b16f15741b3">

        <ds:Transforms>

          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>

          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>

        </ds:Transforms>

        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>[Digest value]</ds:DigestValue>

      </ds:Reference>

    </ds:SignedInfo>
```

```
     <ds:SignatureValue>[Signature value]</ds:SignatureValue>
     <ds:KeyInfo>
       <ds:X509Data>
<ds:X509Certificate>[X509 Certificate]</ds:X509Certificate>
       </ds:X509Data>
     </ds:KeyInfo>
   </ds:Signature>
   <samlp:NameIDPolicy Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress" AllowCreate="true"/>
   <samlp:RequestedAuthnContext Comparison="exact">

<saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtected
Transport</saml:AuthnContextClassRef>
   </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

Note: Inside SAML XML file given identity fields `[Digest value]`, `[Signature value]`, `[X509 Certificate]` shall be replaced by corresponding digest value, signature and X509 certificate for this identity to be authenticated.

CURL COMMAND TO TEST THE OPERATION:

```
curl      --location      --request      POST      'https://ecorridor.iit.cnr.it/asi-
api/v1/eidas/authenticate' \
--header 'Authorization: Bearer [token-value]' \
--header "Content-Type: text/xml;charset=UTF-8" \
--header "SOAPAction:Get" \
--data @authentication_request.xml https://ecorridor.iit.cnr.it/asi-
api/v1/eidas/authenticate?SOAP -v
```

Note: As in previous sections `[token-value]` shall be replaced by the token generated by the Identity Manager component.

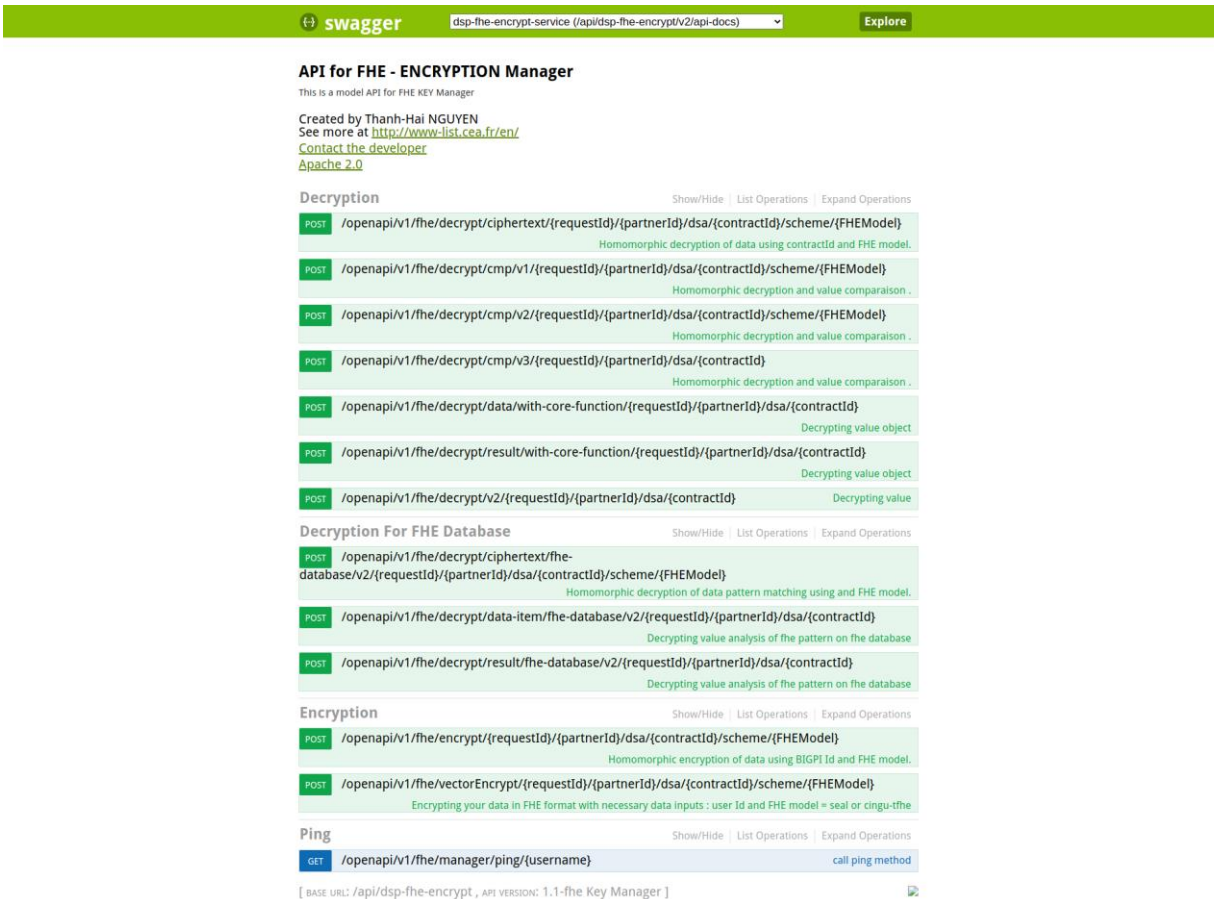
EXPECTED RESPONSE:

When the authentication process is performed successfully the returned response code is 200. Alternatively corresponding error is displayed that is related to server or authentication problem that was identified.

5.4.3. Privacy Aware Interest-Based Service Sharing

This component proposes a service to compute if interest attributes from a passenger match or not with the services provided from a transport company, in two ways. The first one is based on two-party computation, and the second one is based on fully homomorphic encryption (the matching being evaluated in the homomorphic domain as a set intersection between user's interest attributes and a set of available services).

The API depicted in the following figure is devoted to invoke services provided by the fully homomorphic encryption manager, which is also used to provide fully encrypted data to be manipulated by the IAI for other kinds of homomorphic evaluations (see Sections 4.1 and 6.2 of D7.2 for more details). In this figure, each box corresponds to a method that can be invoked in this context, with its high-level description as a green-coloured text.



API for FHE - ENCRYPTION Manager

This is a model API for FHE KEY Manager

Created by Thanh-Hai NGUYEN
See more at http://www-list.cea.fr/en/
Contact the developer
Apache 2.0

Decryption Show/Hide | List Operations | Expand Operations

POST /openapi/v1/fhe/decrypt/ciphertext/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Homomorphic decryption of data using contractId and FHE model.

POST /openapi/v1/fhe/decrypt/cmp/v1/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Homomorphic decryption and value comparaison .

POST /openapi/v1/fhe/decrypt/cmp/v2/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Homomorphic decryption and value comparaison .

POST /openapi/v1/fhe/decrypt/cmp/v3/{requestId}/{partnerId}/dsa/{contractId}
Homomorphic decryption and value comparaison .

POST /openapi/v1/fhe/decrypt/data/with-core-function/{requestId}/{partnerId}/dsa/{contractId}
Decrypting value object

POST /openapi/v1/fhe/decrypt/result/with-core-function/{requestId}/{partnerId}/dsa/{contractId}
Decrypting value object

POST /openapi/v1/fhe/decrypt/v2/{requestId}/{partnerId}/dsa/{contractId} Decrypting value

Decryption For FHE Database Show/Hide | List Operations | Expand Operations

POST /openapi/v1/fhe/decrypt/ciphertext/fhe-database/v2/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Homomorphic decryption of data pattern matching using and FHE model.

POST /openapi/v1/fhe/decrypt/data-item/fhe-database/v2/{requestId}/{partnerId}/dsa/{contractId}
Decrypting value analysis of fhe pattern on fhe database

POST /openapi/v1/fhe/decrypt/result/fhe-database/v2/{requestId}/{partnerId}/dsa/{contractId}
Decrypting value analysis of fhe pattern on fhe database

Encryption Show/Hide | List Operations | Expand Operations

POST /openapi/v1/fhe/encrypt/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Homomorphic encryption of data using BIGPI Id and FHE model.

POST /openapi/v1/fhe/vectorEncrypt/{requestId}/{partnerId}/dsa/{contractId}/scheme/{FHEModel}
Encrypting your data in FHE format with necessary data inputs : user Id and FHE model = seal or cingu-tfhe

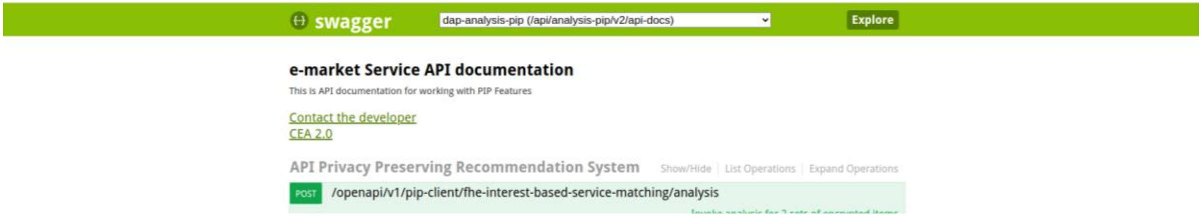
Ping Show/Hide | List Operations | Expand Operations

GET /openapi/v1/fhe/manager/ping/{username} call ping method

[BASE URL: /api/dsp-fhe-encrypt , API VERSION: 1.1-fhe Key Manager]

Figure 5: API for FHE manager

In the same way, the API depicted in the following figure is devoted to run the privacy aware interest-based sharing service itself.



e-market Service API documentation

This is API documentation for working with PIP Features

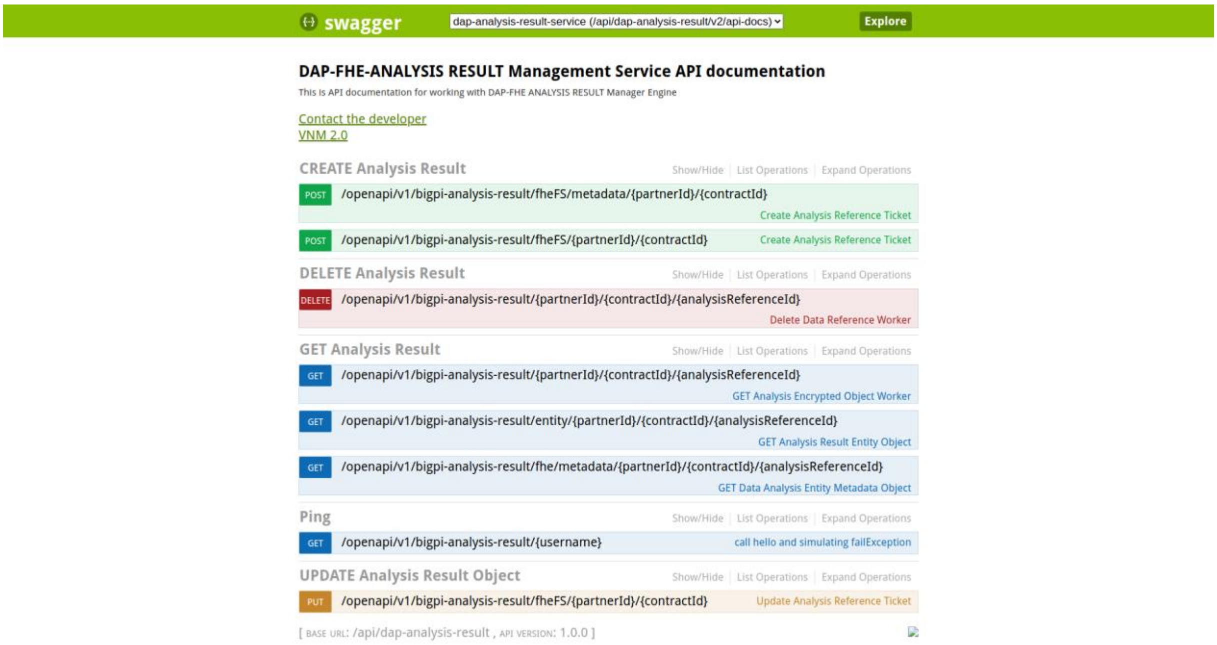
Contact the developer
CEA 2.0

API Privacy Preserving Recommendation System Show/Hide | List Operations | Expand Operations

POST /openapi/v1/pip-client/fhe-interest-based-service-matching/analysis
Invoke analysis for 2 sets of encrypted items

Figure 6: API for interest-based sharing service

Finally, the following API is devoted to get and manage results of homomorphic evaluations:


swagger
dap-analysis-result-service (/api/dap-analysis-result/v2/api-docs)
Explore
Show/Hide | List Operations | Expand Operations

DAP-FHE-ANALYSIS RESULT Management Service API documentation

This is API documentation for working with DAP-FHE ANALYSIS RESULT Manager Engine

Contact the developer
VNM 2.0

CREATE Analysis Result

POST /openapi/v1/bigpi-analysis-result/fheFS/metadata/{partnerId}/{contractId} Create Analysis Reference Ticket

POST /openapi/v1/bigpi-analysis-result/fheFS/{partnerId}/{contractId} Create Analysis Reference Ticket

DELETE Analysis Result

DELETE /openapi/v1/bigpi-analysis-result/{partnerId}/{contractId}/{analysisReferenceId} Delete Data Reference Worker

GET Analysis Result

GET /openapi/v1/bigpi-analysis-result/{partnerId}/{contractId}/{analysisReferenceId} GET Analysis Encrypted Object Worker

GET /openapi/v1/bigpi-analysis-result/entity/{partnerId}/{contractId}/{analysisReferenceId} GET Analysis Result Entity Object

GET /openapi/v1/bigpi-analysis-result/fhe/metadata/{partnerId}/{contractId}/{analysisReferenceId} GET Data Analysis Entity Metadata Object

Ping

GET /openapi/v1/bigpi-analysis-result/{username} call hello and simulating failException

UPDATE Analysis Result Object

PUT /openapi/v1/bigpi-analysis-result/fheFS/{partnerId}/{contractId} Update Analysis Reference Ticket

[BASE URL: /api/dap-analysis-result , API VERSION: 1.0.0]

Figure 7: API for managing results from homomorphic evaluations

6. Conclusions

This document reported the status of the implementation of the E-CORRIDOR Framework at M24. This version will be tested for the Pilots evaluation activities that will be performed till M26 (which is milestone M5 - First Validation of the Pilots and of the E-CORRIDOR platform), where we will get feedback, and will improve and complete the implementation.

We still need to optimise the micro-services application implementation, because not all the services has actually been containerised. This does not prevent the start of the Pilot evaluation phase, because services are available, but this step is necessary to be able to facilitate the deployment of the E-CORRIDOR components in all the Pilots contexts, in particular for the edge deployments.

7. References

Here we provide bibliography references used in the document:

[1] Fundamental modeling concepts (FMC), http://www.fmc-modeling.org (visited on May 16, 2022)

[2] Spring Boot, https://spring.io/projects/spring-boot (visited on May 16, 2022).

[3] Maven, https://maven.apache.org (visited on May 16, 2022).

[4] Curl, https://curl.se (visited on May 16, 2022).