



D6.2

Sharing and Analytics Infrastructures First Maturation

WP6 – Information Sharing and Analytics Infrastructure

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 1.0

Responsible partner: CNR

Editor: Paolo Mori

E-mail address: paolo.mori at iit.cnr.it

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:

P. Mori, O. Oleksii, G. Crincoli, G. Iadarola (CNR), S. Sebastio, S. Daly, P. Sobonski (UTRC), P. Ciampoli (HPE), K. Moulouel (UPEC), M. Ammara (CLEM)

Approved by:

R. Orizio (UTRC), K. Moulouel (UPEC)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	15-feb-2022	P. Mori	CNR	Initial ToC
0.2	16-feb-2022	P. Mori, O. Oleksii	CNR	Section 2 and Section 4: Framework architecture overview and ISI
0.3	21-feb-2022	S. Sebastio	UTRC	Section 3.1.1: AT pilot DSA example
0.4	25-feb-2022	P. Mori, G. Crincoli	CNR	Sections 4.3 and Section 4.4: DMO Toolbox, Obligation Toolbox
0.5	28-feb-2022	S. Daly, S. Sebastio	UTRC	Section 4.3.1: Face anonymization DMO
0.6	5-apr-2022	P. Ciampoli	HPE	Section 3: DSA Lifecycle Infrastructure
0.7	8-apr-2022	P. Mori	CNR	Section 5.1 and Section 5.2: IAI API and Service Usage Control System
0.8	10-apr-2022	P. Sobonski, S. Sebastio	UTRC	Section 5.3: Analytics orchestrator
0.9	11-apr-2022	P. Mori, G. Crincoli	CNR	Section 5.4 and Section 5.5: Analytics Toolbox and Legacy Analytics Engines
0.10	15-apr-2022	O. Oleksii	CNR	Section 4.3.2 and Section 4.4.1 Section : IP Address anonymization DMO and Notification Obligation
0.11	20-apr-2022	K. Moulouel	UPEC	Section 4.3.3: Passenger face extraction DMO
0.12	05-may-2022	P.Mori	CNR	Section 1 and Section 6: Executive Summary, Introduction and Conclusion
0.13	07-may.2022	G. Iadarola	CNR	Section 3.1.3: ISAC pilot DSA example
0.14	10-may-2022	M. Ammara	CLEM	Section 3.1.2: S2C pilot DSA example
0.15	22-may-2022	P. Mori	CNR	Addressed internal reviewers' comments
0.16	24-may-2022	P. Ciampoli	HPE	Addressed internal reviewers' comments
0.17	26-may-2022	S. Sebastio	UTRC	Addressed internal reviewers' comments
1.0	31-may-2022	P. Mori	CNR	Final changes

Executive Summary

Deliverable D6.2 is the second output of Work Package 6, Information Sharing and Analytics Infrastructure. The main aim of this deliverable is to report on the first maturation cycle of three main subsystems of the E-CORRIDOR framework: the **Data Sharing Agreements Lifecycle Infrastructure (DLI)**, in charge of producing and managing the lifecycle of the **Data Sharing Agreements (DSA)**, the **Information Sharing Infrastructure (ISI)**, in charge of providing the privacy preserving data storing and sharing capability, and the **Information Analysis Infrastructure (IAI)**, in charge of providing the data analytics capability. We recall that the E-CORRIDOR framework includes two further subsystems, namely, the Common Security Infrastructure (CSI) and the Advanced Security Infrastructure (ASI), which are not covered by Work Package 6 and hence are not described in this deliverable.

In the first year of the project we designed the internal architectures of the three previously mentioned subsystems and we defined the interactions among the internal components needed to implement the subsystems' functionalities. Then, we mainly focused on the development of the DLI and ISI subsystems, and we delivered a first version of such components implementing some of the main functionalities. These results are described in deliverable D6.1.

In the second year of the project we matured the ISI subsystem by integrating the Data Manipulation Operation Toolbox and the Obligation Toolbox components, thus allowing the integration of Data Manipulation Operations and of Obligations in the E-CORRIDOR framework in an easy way. As a matter of fact, each pilot use case involves its own type of data (e.g., surveillance videos taken in airports or system logs) and has its own privacy requirement (e.g., blur the faces of the passengers in the video or delete part of IP addresses), thus requiring the execution of different Data Manipulation Operations on the data and of different Obligations. By leveraging these components, we integrated in the framework some Data Manipulation Operations and Obligations needed for the pilot use cases, and further ones can be easily added in the next year to accommodate new pilots' needs that could have arisen.

Furthermore, we also matured the IAI subsystem by finalizing the design and developing a first version of the Analytics Toolbox and the Analytics Orchestrator components. The Analytics Toolbox has been integrated in the IAI subsystem. Consequently, new analytics developed and deployed following our specification can be easily integrated in the E-CORRIDOR framework by customizing few configuration files. Additionally, the Analytics Orchestrator allows to define complex analytics as combinations of the existing ones. Again, these components are meant to allow the pilot use cases to define the analytics that are required for their specific scenarios and to easily integrate them in the E-CORRIDOR framework.

Hence, the components integrated in the E-CORRIDOR framework in the second year have the purpose of making the framework highly and easily configurable, to fit the different needs of each pilot use case and potential extension to scenarios outside this project.

Table of contents

- Executive Summary 3
- 1. Introduction 8
 - 1.1. Deliverable Structure 9
 - 1.2. Definitions and Abbreviations 10
- 2. E-CORRIDOR Framework Architecture Overview 12
- 3. Data Sharing Agreement Lifecycle Infrastructure 14
 - 3.1. DSA Editor 14
 - 3.1.1. AT pilot – DSA example 20
 - 3.1.2. S2C pilot – DSA example 22
 - 3.1.3. ISAC pilot – DSA example 24
 - 3.2. DSA API 26
 - 3.3. DSA Mapper 26
 - 3.4. DSA Store 27
 - 3.5. DSA Store Interface 27
- 4. Information Sharing Infrastructure 29
 - 4.1. ISI API 31
 - 4.2. Data Usage Control System 41
 - 4.2.1 Usage Control System 44
 - 4.3. Data Manipulation Operations Toolbox 48
 - 4.3.1. Face Anonymization 50
 - 4.3.2. IP Addresses Anonymization 52
 - 4.3.3. Passenger Face Extraction 55
 - 4.4. Obligations Toolbox 56
 - 4.4.1. Notifications 56
 - 4.5. Bundle Manager 57
 - 4.6. Bundle Store 58
 - 4.7. Bundle Store Interface 58
 - 4.8. Buffer Manager 60
- 5. Information Analytics Infrastructure 62
 - 5.1. IAI API 64
 - 5.2. Service Usage Control System 71
 - 5.3. Analytics Orchestrator 72
 - 5.4. Analytics Toolbox 79
 - 5.5. Legacy Analytics Engines 86
- 6. Conclusion 88

7. References 89

List of Figures

Figure 1 E-CORRIDOR high-level architecture	12
Figure 2 DLI internal architecture.....	14
Figure 3 DSA Editor – “DSAs list” web page	15
Figure 4 DSA Editor – Operation selection for a DSA.....	16
Figure 5 DSA Editor – “New DSA” web page	16
Figure 6 DSA Editor – DSAs templates for Pilots.....	17
Figure 7 DSA Editor – User writing an Authorisation: Selection of a Transportation Sector. 17	
Figure 8 DSA Editor – User writing an Authorisation: Selection of an action.....	18
Figure 9 DSA Editor – User writing an Authorisation: Specify an Organization with free text	18
Figure 10 DSA Editor – Showing a DSA	18
Figure 11 Vocabulary items (extract).....	19
Figure 12 DSA policy for the camera feeds in the AT pilot	22
Figure 13 DSA policy for a user data accessible to both CLEM and NEMI	24
Figure 14 DSA policy for uploading and sharing Automotive data in the ISAC pilot	25
Figure 15 Information Sharing Infrastructure internal architecture.	30
Figure 16 ISI REST API	31
Figure 17 Graphical interface to interact with the E-CORRIDOR framework.....	34
Figure 18 Graphical interface to interact with the E-CORRIDOR framework.....	37
Figure 19 Usage Control Model [PS2004].....	42
Figure 20 Data Usage Control System internal architecture.....	43
Figure 21 DMO Toolbox internal architecture.....	49
Figure 22 Face anonymization pipeline	51
Figure 23 Facial reduction DMO	52
Figure 24 IP Anonymization process	53
Figure 25 Excerpt of a netflow output file	54
Figure 26 Excerpt of a netflow output file after the execution of the IP address anonymization by IP Masking DMO	55
Figure 27 Passenger face extraction pipeline	55
Figure 28 Data Bundle Structure.....	57
Figure 29 Write data to HDFS	60
Figure 30 Information Analytics Infrastructure internal architecture	62
Figure 31 Graphical interface to interact with the E-CORRIDOR framework.....	63
Figure 32 IAI REST API.....	64
Figure 33 IAI REST API: generic API	65
Figure 34 IAI REST API: analytics callback controller	70
Figure 35 Service Level Usage Control System Internal Architecture	71
Figure 36 Analytics Orchestrator in the IAI subsystem.....	74
Figure 37 Example of the steps involved in the workflow execution.....	75
Figure 38 Analytics Orchestrator REST API.....	77
Figure 39 Components in the Analytics Orchestrator as tested in the AT pilot scenario.	78
Figure 40 Script to perform the sequential execution of two analytics.....	79
Figure 41 Analytics Toolbox internal architecture.....	80
Figure 42: Sequence diagram of analytics execution.....	83
Figure 43: Sequence diagram of analytics interruption because of a policy violation	85

1. Introduction

This document is the second output of Work Package 6, Information Sharing and Analytics Infrastructure, due at the end of the second year of the project. The main objective of Work Package 6 “is the development and integration of tools and technologies for both the **Information Sharing Infrastructure (ISI)** and **Information Analysis Infrastructure (IAI)**”, which play a central role in the E-CORRIDOR framework because they implement the two main functionalities of the framework:

- **Privacy-preserving data sharing:** the ISI subsystem allows E-CORRIDOR users to share data among them while preserving their privacy through the enforcement of specific access and usage control policies called **Data Sharing Agreements (DSA)**.
- **Collaborative Analytics execution:** the IAI subsystem allows E-CORRIDOR users to execute data analytics functions on the data shared by any user through the ISI, respecting the constraints expressed in the DSAs linked to such data to protect their privacy.

Moreover, Work Package 6 is also in charge of the design, development and integration in the E-CORRIDOR framework of a third subsystem, the **DSA Lifecycle Infrastructure (DLI)**, which oversees the management of the lifecycle of DSAs, from their creation, through their usage to protect the share data, to their updates, deletion, expiration, or revocation.

This deliverable extends, expands, and supersedes deliverable D6.1 [D6.1] (delivered at the end of the first year) and it describes in detail the status at the end of the second year of the three aforementioned subsystems, DLI, ISI, and IAI. Each of these subsystems corresponds the output of one of the three tasks in Work Package 6, i.e., T6.1: Data Sharing Agreement Infrastructure; T6.2: Data Collection and Usage Enforcement; T6.3 Analytics Infrastructure. Specifically, in the following we describe the functionalities, the internal architecture, and the interactions among the internal components of each of the three subsystems. The interactions among all the subsystems of the E-CORRIDOR architecture (5 in total) to implement the key functionalities of the framework (Create Bundle, Read Bundle, Delete Bundle, Transfer Bundle, Execute Analytics) have been described in deliverable D5.2 [D5.2], and they have not undergone relevant updates in the second year of the project.

The three subsystems, DLI, ISI, and IAI, are derived from the C3ISP project (H2020-DS-2015-1, Collaborative and Confidential Information Sharing and Analysis for Cyber Protection, C3ISP, GA#700294). While C3ISP was focussed on sharing and analysing a very specific set of data and analytics for describing and analysing Cyber Threat Information (CTI), E-CORRIDOR focusses on the multimodal transportation scenario and it generalizes these concepts and ideas to a fully-fledged framework for sharing and analysing any kind of data, any kind of analytics, and also enabling analytics at the edge.

In the first year of the project we designed the updated internal architectures of the three subsystems, and we defined the interactions among the internal components of the three subsystems to implement the subsystems’ functionalities. Then, we mainly focused on the development of the DLI and ISI subsystems, and we delivered a first version of the components implementing some of the main functionalities. These results are described in deliverable D6.1.

In the second year of the project we matured both the ISI and the IAI subsystems: we focused on the Data Manipulation Operation Toolbox and Obligation Toolbox components of the ISI subsystem and on the Analytics Toolbox and Analytics Orchestrator components of IAI subsystem. These components have the purpose of making the E-CORRIDOR framework

highly and easily configurable, enabling the framework to be considered for a large number of scenarios, starting from the pilot use cases of the project. As a matter of fact, the Data Manipulation Operation Toolbox allows to define operations meant to preserve the privacy of the data to be shared through the framework as external services, and to easily integrate such services in the E-CORRIDOR framework through some configuration files. The idea is that specific Data Manipulation Operations (DMO) are developed, deployed, and integrated in the E-CORRIDOR framework for each pilot use case, to accommodate the different privacy requirements of each of them. Obviously, each time a new DMO is integrated in the E-CORRIDOR framework, the vocabulary used to express the DSAs must be updated to include such DMO among the list of the available ones. Similarly, the Obligation Toolbox allows to define a set of operations to be executed when data is accessed (e.g., notification to the producer of such data) as external services as well (again, developed for the pilot use cases' needs, but following our specifications) and to integrate such services in the E-CORRIDOR framework in the same way as the Data Manipulation Operations. The Analytics toolbox, instead, allows to implement the analytics operations that will be executed on the data shared within the E-CORRIDOR framework as external services as well, while the Analytics Orchestrator allows to further extend the analytics provided by the framework by creating new analytics as a composition of existing ones. These components made the E-CORRIDOR framework flexible, configurable, and adaptive to a very large number of distinct scenarios, involving any kind of data and analysis. Furthermore, in the second year we designed, developed and integrated in the framework some Data Manipulation Operations and Obligations necessary to the pilot use cases. Also several analytics have been designed and integrated in the E-CORRIDOR framework following the previously described approach. These analytics are described in deliverable D7.2 [D7.2].

The following of this document focuses on the description of the DLI, ISI, and IAI subsystems, focussing on the new components and on their integration in the E-CORRIDOR framework. However, for the sake of completeness, this document also provides a description of all the other components of the three subsystems, even if their architectures, features, and functionalities have not been undergone major changes w.r.t. the version delivered at the end of the first year.

1.1. Deliverable Structure

This document is structured as follows:

- Chapter 2 gives a brief and high-level overview of the overall architecture of the E-CORRIDOR framework.
- Chapter 3 describes the internal architecture of the DLI subsystem, explaining the interfaces exposed by the subsystem for operating on DSAs, and the interactions among the internal components to implement the subsystem functionality.
- Chapter 4 describes the ISI subsystem, giving a detailed description of the interfaces provided by the subsystem to the E-CORRIDOR users and of all the components of the subsystem. We give a detailed description of the Data Manipulation Operations Toolbox and of the Obligations Toolbox, which have been designed, developed, and integrated in the framework in the second year of the project, as well as some Data Manipulation Operations and Obligations which have been integrated in the framework exploiting such toolboxes.

- Chapter 5 describes the IAI subsystem, giving a detailed description of the interfaces provided by the subsystem to the E-CORRIDOR users and of all the components of the subsystem. We give a detailed description of the Analytics Toolbox and of the Analytics Orchestrator, which have been designed and developed in the second year of the project in order to allow the integration of any analytics in the E-CORRIDOR framework. The detailed description of the specific analytics that will be integrated exploiting the Analytics Toolbox is given in deliverable D7.2.

1.2. Definitions and Abbreviations

Table 1 lists the abbreviations that have been used in this document.

Table 1 Abbreviations used in the document

Term	Meaning
AM	Attribute Manager
API	Application Programming Interface
ASI	Advanced Security Infrastructure
CH	Context Handler
CNL	Controlled Natural Language
CRUD	Create, Read, Update, Delete
CSI	Common Security Infrastructure
CTI	Cyber Threat Information
DC	Decision Combiner
DSA	Data Sharing Agreement
DLI	DSA Lifecycle Infrastructure
DMO	Data Manipulation Operation
FHE	Full Homomorphic Encryption
HDFS	Hadoop Distributed File System
IAI	Information Analytics Infrastructure
IDS	Intrusion Detection System
ISI	Information Sharing Infrastructure
JAR	Java ARchive
JSON	JavaScript Object Notation
MRH	Multi Request Resource Handler
OWL	Ontology Web Language

PDP	Policy Decision Point
PIP	Policy Information Point
SMS	Short Messages Service
SQL	Structured Query Language
UCON	Usage Control
UPOL	Usage Control Policy
VDL	Virtual Data Lake
XACML	eXtensible Access Control Markup Language

2. E-CORRIDOR Framework Architecture Overview

As explained in deliverable D5.4 [D5.4] the final version of the overall architecture of the E-CORRIDOR framework has not undergone to relevant changes with respect to the version presented at the end of the first year, which is described in deliverable D5.2. As a matter of fact, the main changes have been done at the level of the internal component of each subsystem, as will be detailed in the rest of this document. The high level architecture is shown in Figure 1, and includes 5 subsystems: the DSA Lifecycle Infrastructure (DLI), the Information Sharing Infrastructure (ISI), the Information Analytics Infrastructure (IAI), the Common Security Infrastructure (CSI) and the Advanced Security Infrastructure (ASI).

This deliverables describes the DLI, the ISI, and the IAI subsystems, and this section gives a very brief and high-level description of the main interactions initiated by the DLI, by the ISI and by the IAI with the other subsystems. A detailed description of the interactions among all the five subsystems of the E-CORRIDOR architecture to implement the workflows of some functionalities of the framework (Create Bundle, Read Bundle, Delete Bundle, Transfer Bundle, Execute Analytics) have been described in deliverable D5.2, and they have not modified in the second year of the project.

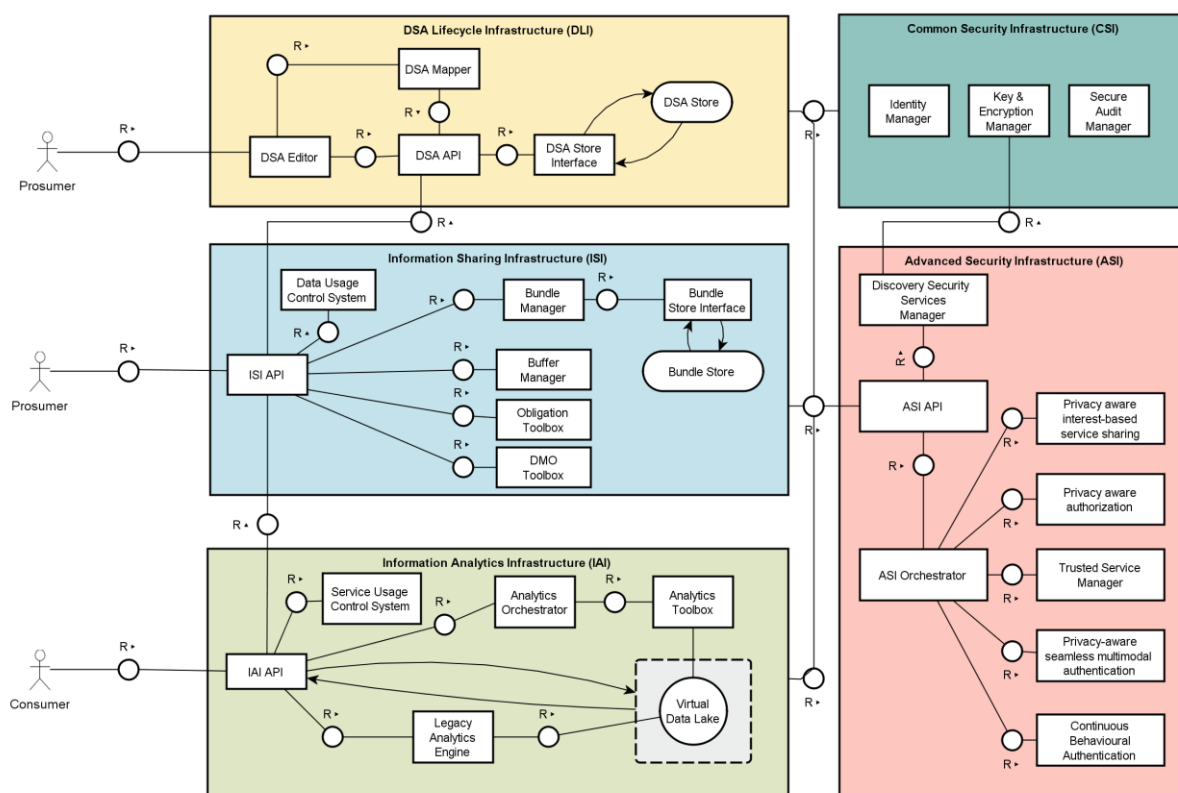


Figure 1 E-CORRIDOR high-level architecture

DSA Lifecycle Infrastructure: this subsystem allows data producers to manage the DSAs they use to protect the data they produce. The DLI subsystem provides a graphical and user-friendly interface, the DSA Editor, to create new DSA, to visualize and update existing ones, and to invalidate or even delete them. The DSAs created through the editor are then translated in executable format in order to be paired with the data and enforced by the ISI subsystem. As described in details in deliverable D5.4 (single sign on) user authentication for this subsystem

is implemented by mean of a Keycloak [KEY] service, which is deployed as part of the CSI subsystem.

Information Sharing Infrastructure: this subsystem is in charge of managing the data in the E-CORRIDOR framework, ensuring both their secure storage and their privacy preserving sharing, following the Usage Control model. In particular, to protect data confidentiality when they are at rest, the ISI embeds data in cryptographic containers (called Data Bundles) before being stored on the storage system. Instead, for protecting data privacy when they are shared with other users to perform collaborative analytics, the ISI also embeds DSAs in the Data Bundles and, every time that a user requests to execute a collaborative analytics on a Data Bundle, the corresponding DSA is enforced to check whether such user has the right to exploit such data to perform analytics. Moreover, the ISI subsystem also allows the integration of privacy preserving operations to following a plugin approach. Each of these operations is executed on the data when required by the DSA before executing the analytics.

As described in details in D5.4 authentication of users with single sign on for this component is implemented exploiting the Keycloak service deployed as part of the CSI subsystem. The Keycloak service will be used also as user attributes provider by the Data Usage Control Service. The CSI subsystem is invoked also to retrieve the keys for encrypting and decrypting the data embedded in Data Bundles.

The ISI subsystem invokes the DLI subsystem to retrieve a DSA when a Data Bundle is created, and every time that a Data Bundle is being accessed (for reading it or for the execution of an analytics), in order to check whether the associated DSA is still valid, i.e., it has not been invalidated by the creator, and it has not been superseded by a new version.

Finally, the ISI subsystem invokes the IAI to request to interrupt the execution of a running analytic when a violation of the DSA of one of the data that are being exploited for such analytic occurs while the analytic is still in progress (i.e., results have not been released yet).

Information Analytics Infrastructure: this subsystem manages the execution of the collaborative analytics on the data shared through the ISI subsystem. Data consumers interact with the IAI to request the execution of collaborative analytics on a given set of Data Bundles, which is typically defined using a query on the metadata paired with the Data Bundles. Hence, the IAI invokes the ISI subsystem to search and retrieve the Data Bundles, which are used for executing the requested analytics. When the analytics execution has been completed, the IAI subsystem interacts with the ISI subsystem to create a new Data Bundle embedding the results of the execution.

Moreover, this subsystem as well, exploits the Keycloak service deployed as part of the CSI subsystem for (single sign on) user authentication.

Finally, the three previous subsystems interact with the CSI subsystem for exploiting the E-CORRIDOR secure auditing facilities.

3. Data Sharing Agreement Lifecycle Infrastructure

This chapter describes the subsystem DSA Lifecycle Infrastructure (DLI) with its components and functions as already introduced in deliverable D6.1. The DLI subsystem, which is in charge of DSAs management, has evolved its internal components that now are built with micro-services as depicted in Figure 2:

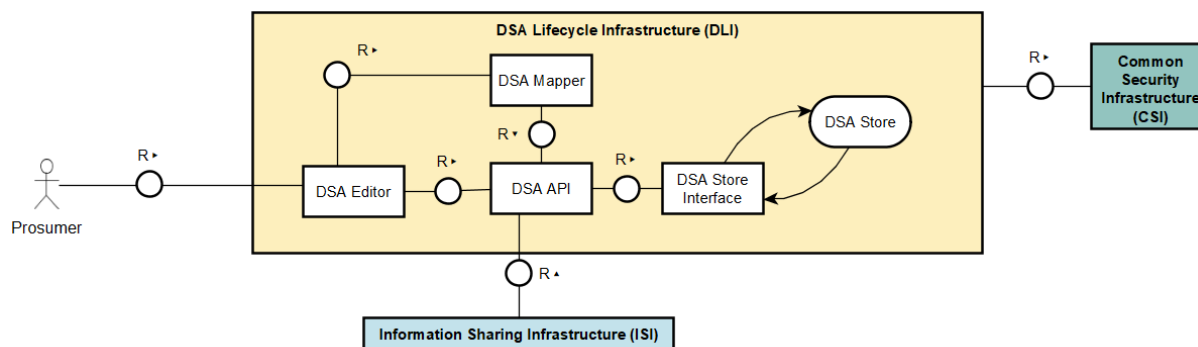


Figure 2 DLI internal architecture

Components are:

- DSA Editor: Web User Interface for the management of DSAs.
- DSA API: front-end to call DSA functions of the platform.
- DSA Mapper: translator from Controlled natural Language (CNL) statements to programmatic instructions.
- DSA Store: the database for storing DSAs.
- DSA Store Interface: programmatic interface to the DSA Store.

In the next paragraphs we describe the implementation of these components, which is the same for both Development and Test environments.

The plan is to implement all components with docker containers [DOC2021].

3.1. *DSA Editor*

The DSA Editor component is composed by four modules:

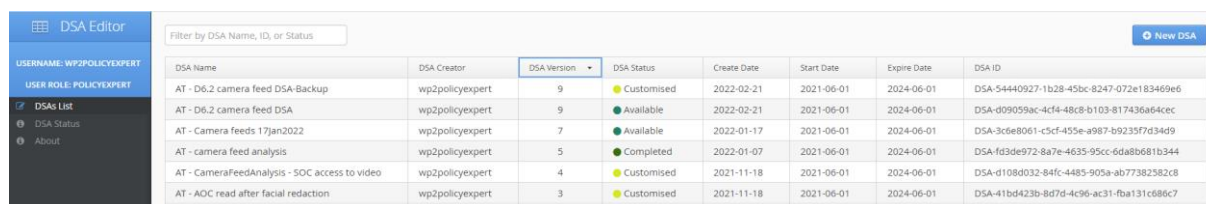
- dsa-editor-frontend: the frontend part of the Editor (the Web GUI).
- dsa-editor: the management DSA tool (or authoring tool).
- dsa-store-mariadb: the internal database where are stored users and association between DSAs and users.
- dsa-vocabulary: the internal vocabulary, in RDF format with extension .owl, containing the ontology for E-Corridor DSAs

The frontend part manages user access, DSAs listing and in general web visualization and user interaction and for this it uses a database implemented with mariadb software for storing users and some DSAs metadata to faster retrieving of DSAs.

The DSA authoring tool is the core of the tool for managing DSAs: CRUD (Create, Read, Update, Delete) operations on each DSA and invoking of mapper tool. This part of applications also reads ontology (that is stored in vocabulary in the form of a OWL file [AVH2003]) that is the base for the DSA rules constructed with the aim to guide the user to the writing of sentences. This is further explained in the following sections with some examples.

Access to the DSA Editor is with the link <https://dsamgr2.iit.cnr.it/DSAEditor/> and with an account identified by user and password that will be mapped with a Keycloak account (integration is in progress). We assigned one different user for each pilot and we have also some service users for DSAs template creation (see deliverable D6.1) and for tests. DSA templates are used as a base for new DSA and to associate them to a specific vocabulary version (e.g., development vocabulary, demo vocabulary, etc.).

When user makes login to DSA Editor can see a list of DSAs that she/he has created (or an empty list if there are no DSAs associates to the account).



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-d090595ac-4cf4-48cb-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-fd3de972-9a7e-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-fba131c688c7

Figure 3 DSA Editor – “DSAs list” web page

Here user can create a new DSA with the selection button on the top-right corner or position the mouse on a specific DSA and then select one of the possible operation that appear on the right side (Show, Edit, Copy, Raw, Map, Delete, Revoke):

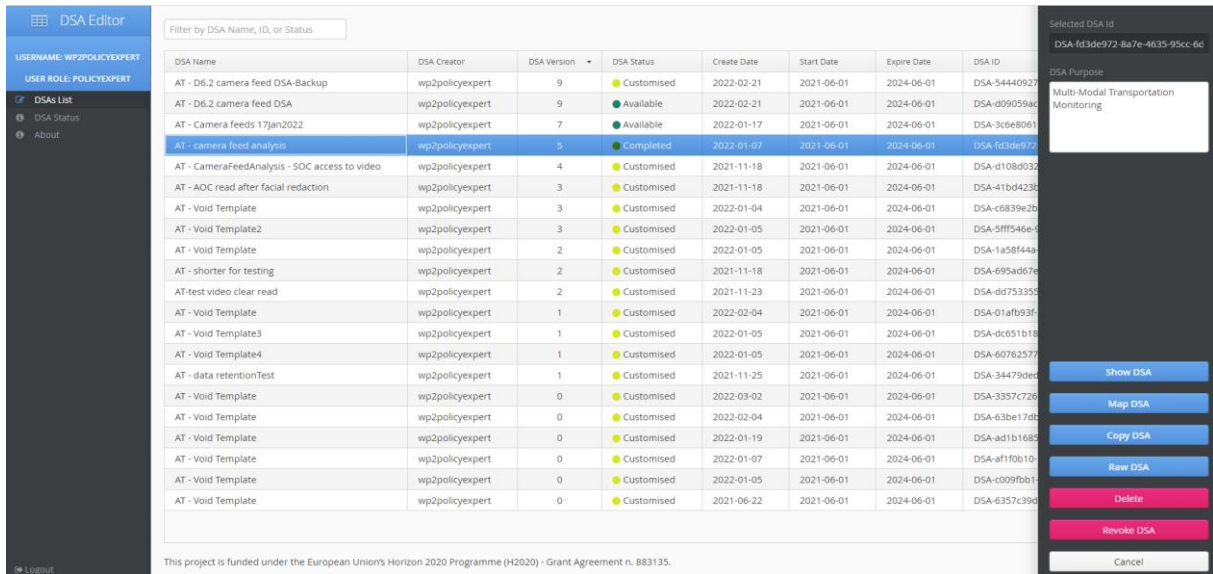


Figure 4 DSA Editor – Operation selection for a DSA

When user select new DSA, first appear a list of template selection and then a web page where user can compile metadata fields for the DSA (Title, Description, Validity, Parties)

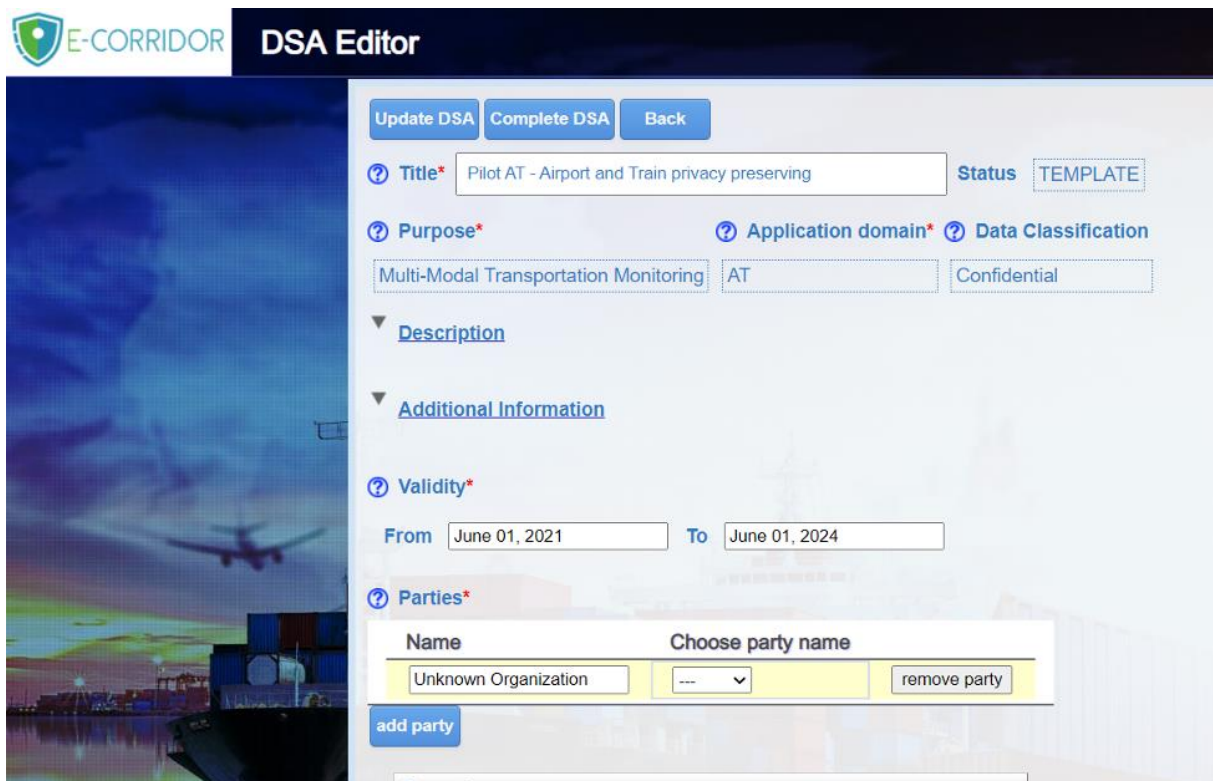


Figure 5 DSA Editor – “New DSA” web page

Some data are already prepared coming from template (Purpose, Additional information) and DSA status.

In the DSA Editor we defined three “Application Domain” (AT, S2C, ISAC), one for each Pilot, and we made three corresponding templates (named “*Pilot-name* – Void Template”) to allow each of them to create DSA with specific Application Domain.

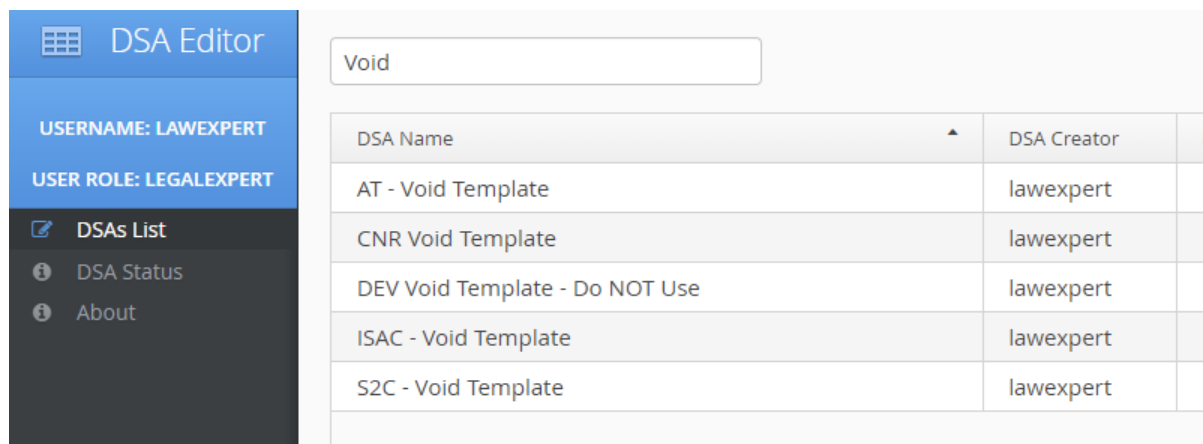



Figure 6 DSA Editor – DSAs templates for Pilots

Then user can write DSA rules (Authorization, Obligation, Prohibition) in the section under metadata selecting the  button under the correspondent part.

We are now going to describe in detail the steps required for a user to create a READ authorization, which is then going to be further expanded on a pilot scenario in Section 3.1.1:

1. first user selects a Subject then a property of this Subject, “hasTransportationSector” picking from a menu the specific Sector (in this case “Aviation”)



Figure 7 DSA Editor – User writing an Authorisation: Selection of a Transportation Sector

2. adds another property to same Subject selecting “hasOrganization”
3. chooses action, “Read”, from a menu



Figure 8 DSA Editor – User writing an Authorisation: Selection of an action

4. and fills the user free text with the specific name of the Organization in this case “ADP”

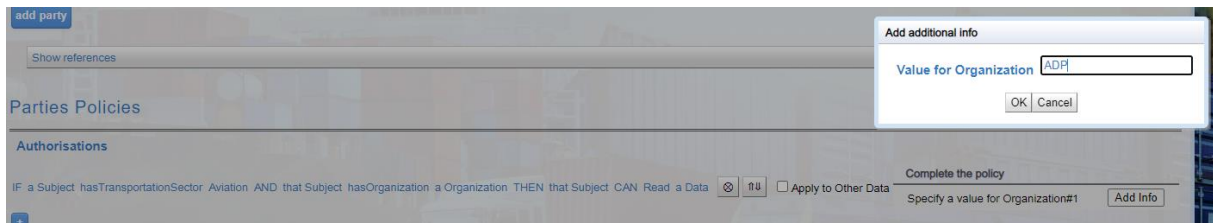


Figure 9 DSA Editor – User writing an Authorisation: Specify an Organization with free text

When user updates DSA and selects Show of that DSA, the Editor tool displays in the Policies section the READ Authorization that user wrote:

Type	Policies
AUTHORIZATION	IF a Subject hasTransportationSector Aviation AND that Subject hasOrganization a Organization(ADP) THEN that Subject CAN Read a Data

Figure 10 DSA Editor – Showing a DSA

Writing of this sentence is made with the constraint of a vocabulary that is explicitly designed for the domain that in our case includes terms related to Multi Modal Transport environment and project pilot contexts.

Here an extract of actual vocabulary with meaningful items:

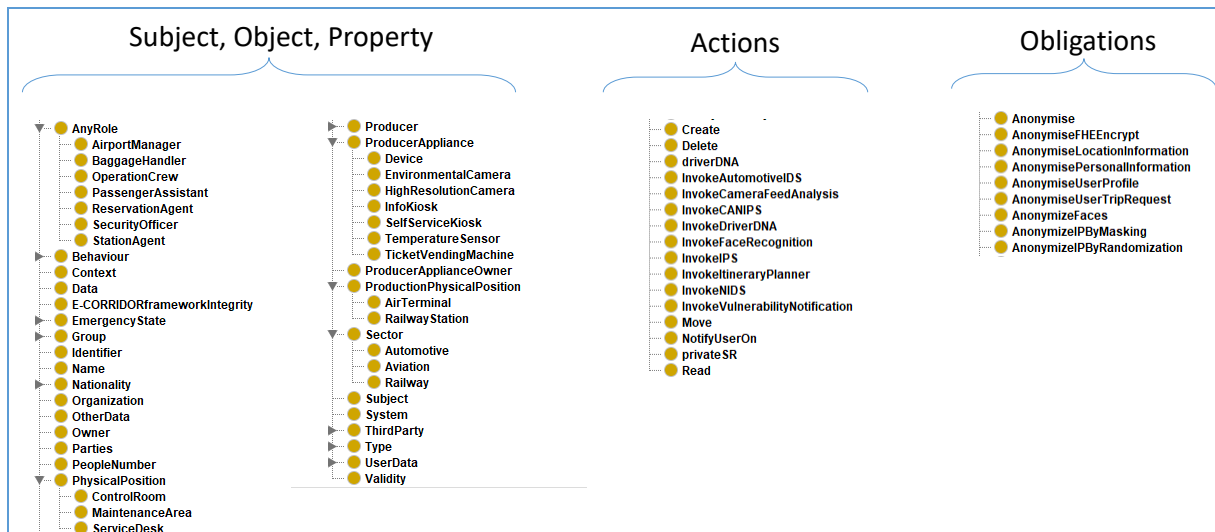


Figure 11 Vocabulary items (extract)

Vocabulary has items hierarchy organised under two principal entities: Term and Action.

Under Terms are defined all Subjects (Subject, System) and Objects (Data) of policies and some Properties (TransportationSector, Organization, AnyRole,...) that allow properties characterization of them.

Vocabulary items are defined following pilots requirements and actual vocabulary reflects needs emerged until now for policies writing from AT, S2C and ISAC contexts.

Vocabulary can have different versions. Version for a single DSA is fixed at creation time and is based on the underlying template and cannot be changed. Vocabularies version remains available to allow reading of DSA already created. Needs for a new version arise when there are meaningful changes to a vocabulary that impacts in particular items already existing (e.g. deleting, renaming or type changing for items). If there was only one vocabulary, after changes some DSAs could be unreadable and no more usable from Editor.

Different versions are also useful for testing a new ontology or for coexistence of different environments (development, test or demo where you don't want make changes).

Vocabulary is in a first version, refined during this 2nd year of project, that allow pilots to write their policy rules and it will be enriched during project prosecution.

The vocabulary is currently available in the development environment at (https://dsamgr2.iit.cnr.it/vocabularies/ecorridor_vocabulary_0.1.owl#).

In next paragraph 3.1.1 is presented, as an example of functionality, a policy related to AT Pilot written using the Editor Tool.

3.1.1. AT pilot – DSA example

Scenario: the airport and train (AT) pilot aims at demonstrating privacy-preserving solutions supporting passengers choosing multi-modal journeys. For security and performance reasons several environmental cameras are deployed in the airport terminal and train station. The captured videos are analysed by machine learning (ML) algorithms to detect anomalies or extract performance metrics. Examples of these two types of applications are the respect of Covid-19 physical distancing rule (about 1 meter from others [WHO22]) for the former, and time spent near vending machines for the latter. In addition to being regularly and automatically processed by ML solutions, in case of emergency, the same videos may need to be accessed also by the staff of the security operation centre operating either remotely or in the control room (in the first case an email notification must be sent). To avoid any improper use from the operators and at the same time to preserve the privacy of all the passengers accessing the transportation facilities, face anonymization techniques are highly appreciated.

Moreover, since the videos record the passengers, these contain sensitive data. According to the GDPR principles, data must be deleted once the original purpose for collection does not apply. In the depicted scenario, this is represented by the duration of the passengers' travels and can be translated in a data retention policy.

DSA: The above defined scenario highlights the need for regulating different aspects pertaining the data control policy: creation, access, use, retention. Such requirements are realized by means of three authorizations and two obligation clauses.

In the following DSA policy, in italic are marked the values for the corresponding attribute. If the value is entered as free text, it is in quotes. Conversely, it is selected from a set of predefined options.

Creation (Authorization):

```
IF aSubject hasOrganization "ADP" OR thatSubject hasOrganization "SNCF" AND
thatSubject hasTransportationSector Aviation OR thatSubject hasTransportationSector
Railway AND aData hasProducer ADP OR thatData hasProducer SNCF AND thatData
hasType Video AND thatData hasProducerAppliance EnvironmentalCamera AND thatData
hasProductionPhysicalPosition AirTerminal OR thatData hasProductionPhysicalPosition
RailwayStation AND thatData hasProducerApplianceOwner "Genetec" THEN thatSubject
CAN CREATE thatData
```

Note: Genetec is a provider of IP-based security solutions and exemplify a third party working in the AT pilot acting as data producer.

Access (Authorization):

IF aSubject hasRole *SecurityOfficer* AND thatSubject hasTransportationSector *Aviation* OR thatSubject hasTransportationSector *Railway* AND thatSubject isMemberOf *SecurityOperationCenter* AND aData hasType *Video* AND thatData hasProductionPhysicalPosition *AirTerminal* OR thatData hasProductionPhysicalPosition *RailwayStation* AND thatData hasProducerAppliance *EnvironmentalCamera* AND thatData hasProducer *ADP* OR thatData hasProducer *SNCF* AND aContext hasContextEmergencyState *emergency* THEN thatSubject CAN READ thatData

Access-restriction (Obligations):

AFTER aSubject READ aData THEN aSystem MUST *AnonymizeFaces* thatData

AFTER aSubject READ aData THEN IF thatSubject NOT hasPhysicalPosition *ControlRoom* THEN aSystem MUST *notifyUserOn* “(email: soc_control@adp.com)” thatData

Note: AnonymizeFaces is a Data Manipulation Operation (DMO) available in the DMO toolbox and discussed later in Sec 4.3.1. NotifyUserOn is an obligation available in the Obligations toolbox (discussed in Sec **Error! Reference source not found.**) and used here to notify the security control centre by email in case the access to the videos is performed remotely.

Use (Authorization):

IF aData hasType *Video* AND thatData hasProductionPhysicalPosition *AirTerminal* OR thatData hasProductionPhysicalPosition *RailwayStation* AND thatData hasProducerAppliance *EnvironmentalCamera* AND thatData hasProducer *ADP* OR thatData hasProducer *SNCF* THEN aSubject CAN *InvokeCameraFeedAnalysis* thatData

Note: CameraFeedAnalysis is a data analytics available in the analytics toolbox and discussed in D7.2.

Data retention (revocation policy):

It is defined with a cancellation period expressed in days. In the editor, this option is available in the bottom right part of the interface.

DSA in the editor: The above described DSA policy has been implemented through the E-CORRIDOR DSA editor. The screenshot in **Error! Reference source not found.** represents such a policy (UUID: d09059ac-4cf4-48c8-b103-817436a64cec.xml).

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

Purpose: Multi-Modal Transportation Monitoring **Application domain:** AT **Data Classification:** Confidential

Description: this policy implements the scenario for the AT pilot and the camera feeds as presented and discussed in D6.2

Additional information:
UUID: DSA-e09059ac-4cf4-48c8-b103-817436a64acc.xml
Vocabulary URI: https://dsamg2.il.cnr.it/vocabularies/ecorridor_vocabulary_0.1.owl#

Validity: From: May 31, 2021 To: May 31, 2024

Parties: ADP, SNCF

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(Genetic) 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 SecurityOperatorCenter 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(parameters optionProc_control@adp.com) that Data

General Policies: DenyAll, DenyAll, DeleteInSpecifiedPeriod

Expiration Policy: Period In days: N/A

Update Policy: Period In days: N/A

Revocation Policy: Period In days: 7

Figure 12 DSA policy for the camera feeds in the AT pilot

3.1.2. S2C pilot – DSA example

Scenario: The Smart city and Carsharing Pilot (S2C Pilot) aims to offer a seamless and secure traveller journey from the account creation to the usage of the account data to book trips. During the subscription (i.e.: account creation) the traveller must be able to define who has access to the personal data amongst the mobility service providers.

DSA:

The following example explains how the DSA would enforce the traveller's choices over his/her own data. DATA= traveller's personal information submitted through the eWallet web app shall be accessible to the selected Mobility Service Providers only. Furthermore, when such a Mobility Service Provider retrieves the traveller's data, data fields which aren't used nor needed by the Mobility Service Provider are anonymized. This way, we respect the data minimization principle.

In this case we have 3 DSAs, one for CLEM & PLD, one for CLEM only, one for PLD only.

Creation: This part is common among the 3 DSAs

AUTHORIZATION : IF a Subject hasId a Identifier(eWallet Web App) AND a Subject hasPhysicalPosition a PhysicalPosition THEN a Subject CAN Create a Data

Authorization and Obligations: This part is different according to the traveller's choice**DSA CLEM & PLD**

AUTHORIZATION : IF a Subject hasId a Identifier(eWallet Web App) AND a Subject hasPhysicalPosition a PhysicalPosition THEN a Subject CAN Create a Data

AUTHORIZATION : IF a Subject hasOrganization a Organization(CLEM) OR that Subject hasOrganization a Organization(PLD) THEN that Subject CAN Read that Data

OBLIGATION : IF that Subject hasOrganization that Organization THEN a System MUST AnonymisePersonalInformation that Data

DSA CLEM

AUTHORIZATION : IF a Subject hasOrganization a Organization(CLEM) THEN that Subject CAN Read that Data

DSA PLD

AUTHORIZATION : IF a Subject hasOrganization a Organization(PLD) THEN that Subject CAN Read that Data

OBLIGATION : IF that Subject hasOrganization that Organization THEN a System MUST AnonymisePersonalInformation that Data

DSA in the editor: The described DSA policy has been implemented through the E-CORRIDOR DSA editor. The screenshot in Figure 13 represents the policy in the case of a traveller choosing to both CLEM and PLD for his/her data (UUID: DSA-183b1d8c-b377-4800-8dac-f92d43628d5c)

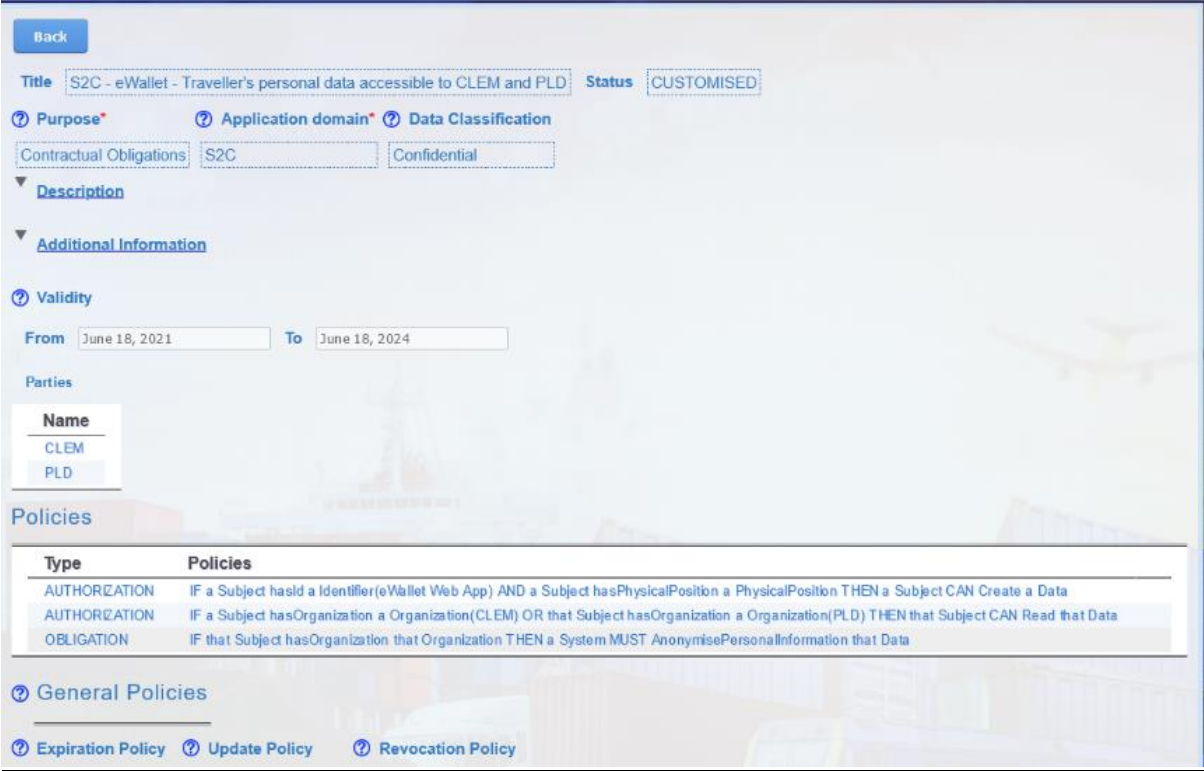


Figure 13 DSA policy for a user data accessible to both CLEM and NEMI

3.1.3. ISAC pilot – DSA example

Scenario: The Information Sharing and Analytics Centre (ISAC) pilot aims to collect, analyse and disseminate actionable threat information to the framework partner and registered users, and also provide tools to mitigate risks and enhance resiliency. The ISAC provides a web platform with graphs and statistics from public data sources regarding cybersecurity threats and also allows to upload users own data to share information with other entities, run analytics and collect results.

The users interacting with the ISAC portal came from different business sectors, and have different goals: while a generic user may be interested in reading the public data available only, a member of a transportation sector may be interested in uploading data and running custom and sound analysis on his dataset. Also, the latter could be interested in sharing the dataset or the analytics result with partners of the same transportation sector and keeping the data confidential. On the other hand, the ISAC supports data sharing within the users' community in terms of data aggregation, thus, the data owner needs to have control over the data sharing policy applied to its data when uploading on the ISAC portal.

In this regard, it plays a crucial role to implement several layers of data and usage control over the ISAC uploaded data, with regard to the sharing and the privacy policies.

DSA: The scenario highlights the need for different data access and usage over the ISAC uploaded data. Each ISAC tool/service requires different data access, but we can group the requirements into three data sharing policies: public data (accessible by all the users), private data (accessible only by the data owner), and private within a transportation sector and/or an organization (accessible only by members of the same transportation sector/organization).

In the following example, the DSA implements a policy for a data that can be shared only within the Automotive sector partners, and specifically by a subset of organizations.

In the DSA policy, in italic are marked the values for the corresponding attribute. If the value is entered as free text, it is in quotes. Conversely, it is selected from a set of predefined options.

Creation (Authorization):

IF a Subject hasOrganization "CNR" OR that Subject hasOrganization "HPE" AND that Subject hasTransportationSector *Automotive* THEN that Subject CAN Create a Data

Access (Authorization):

IF a Subject hasOrganization "CNR" OR that Subject hasOrganization *HPE* AND that Subject hasTransportationSector *Automotive* THEN that Subject CAN Read that Data

DSA in the editor: The described DSA policy has been implemented through the E-CORRIDOR DSA editor. The screenshot in Figure 14 represents such a policy (UUID: DSA-3b2407bf-3578-489b-8b94-c0c7012308f5).

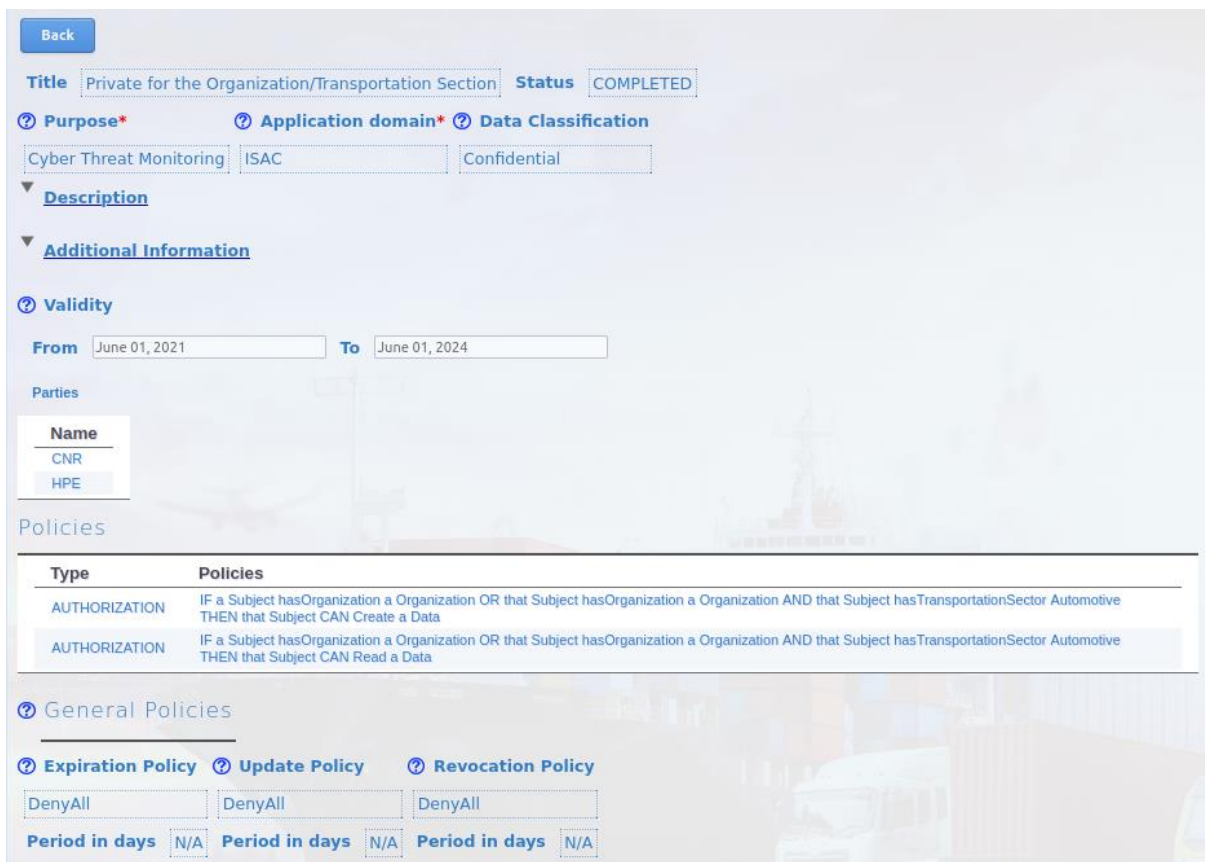


Figure 14 DSA policy for uploading and sharing Automotive data in the ISAC pilot

3.2. *DSA API*

DSA management can be made in interactive mode with DSA Editor Tool or, for some operations, with REST APIs used internally from the components.

Until now, no new APIs were needed compared to the set already listed in deliverable D6.1, reported in Table 2 for simplicity:

Table 2 – DSA APIs

API	Input	Output
Create DSA /createdsa	DSA identifier, dsa .xml file, userid	new DSA in the store
Delete DSA /deletedsa/{dsaid}	DSA identifier	Operation result (success or failure)
Retrieve DSA /getDSA/{dsaid}	DSA identifier	DSA .xml file
Retrieve the status of a DSA /getstatus	DSA identifier	DSA status
Mapping of a DSA /mapdsa/{dsaid}	DSA identifier	Operation result (success or failure)
Revoke a DSA /revoke/{dsaid}	DSA identifier	Operation result (success or failure)
Update DSA /updatedsa	DSA identifier, updated DSA data	Operation result (success or failure)
Update DSA status /updatestatus	DSA identifier	DSA status changed

3.3. *DSA Mapper*

The DSA Mapper is the component in charge of translating the policies defined through the DSA Editor, which are specified in CNL language, into automatically enforceable ones. Once a DSA has been considered in its final form, a translation from CNL to an executable format is required to allow the policy to be enforced by the Data Usage Control System (see Section 4.2). The Data Usage Control System works by exploiting the UPOL language

[DLMMM2018], an extension of the XACML language [XACML2013] that supports usage control features.

The set of policies of a DSA must be able to regulate a plethora of aspects related to all E-CORRIDOR Pilots, e.g., organisational policies, security requirements, privacy regulations, and so on. This will be done without making specific assumptions about the application model. This makes the work of the DSA mapper very difficult, if not intractable in the general case.

The DSA Mapper works in two phases:

1. As first step, the DSA Mapper automatically processes the pilot vocabularies in order to learn all the terms that may be present in a policy. Terms collected into the vocabulary are referred to subjects, objects, actions and properties that allow the user to write the policy by using the DSA Editor. Each term in the vocabulary corresponds to an OWL object. To be completely processed, each term needs to be labelled in order to understand if it has to be evaluated only at the time of the access request (Access Control) or is to be continuously evaluated during the access time (Usage Control).
2. As second step, the DSA Mapper translated both the syntax and semantics of the CNL policies by associating each CNL construct to a UPOL one. Thus, the DSA Mapper is able to identify the following main elements of a policy:
 - a. The subject that requests the access. A subject has one or more attributes.
 - b. The object, that is mainly the resource element. Usually it is a data, a service or a component of the system. A resource has one or more attributes.
 - c. The action that has to be enforced. Actions have one or more attributes.
 - d. The environment that may optionally provide additional information.

3.4. *DSA Store*

DSA Store is the repository of DSAs (.xml files). The DSA Store is implemented with database software MongoDB.

3.5. *DSA Store Interface*

For the sake of completeness, Table 3 lists the DSA Store APIs, internally used from DSA API component, presented in deliverable D6.1:

Table 3 – DSA Store APIs

API	Input	Output
Create DSA /createDSA	DSA metadata, policies and properties	DSA identifier

Delete DSA /deleteDSA/{dsaid}	DSA identifier	Operation result (success or failure)
Read DSA /readDSA/{dsaid}	DSA identifier	DSA .xml file
Search DSA /searchDSA/{longResultFlag}	Search criteria	A set of DSA Identifiers corresponding to the matching DSAs
Update DSA /updateDSA	DSA identifier, updated DSA data	Operation result (success or failure)

To this point, no major changes were required to the set of APIs since the initial set covered all the needs and no further functionalities were needed.

4. Information Sharing Infrastructure

The Information Sharing Infrastructure (ISI) subsystem is in charge of managing data in the E-CORRIDOR framework, ensuring both their secure storage and their privacy preserving sharing, following the Usage Control model. The ISI implements the secure storage by providing the support for encrypted storage of data, and the privacy preserving data sharing by pairing a DSA with each piece of data, and by guaranteeing the enforcement of such DSA to regulate the sharing of such piece of data with the other users of the framework.

As a matter of fact, to protect data when they are at rest, data are embedded in a cryptographic container (called Data Bundle) before being stored on the storage system. The Data Bundle includes the DSA paired with the data, and some metadata, which are meant to be public and, consequently, they are stored in clear (not encrypted) in the bundle and they are used in the search operation.

When a user of the E-CORRIDOR framework requests to access the data embedded in a Data Bundle for performing a given operation (e.g., READ, DELETE, or an analytics), the DSA in the Data Bundle is evaluated to decide whether the user is allowed or not to perform such operation. While the data extracted from a Data Bundle is in usage (for instance, an analytics involving the data is running), the related DSA is continuously evaluated over time, according to the Usage Control model, to decide whether the execution of such analytics can go on or it must be interrupted because of a policy violation. The continuous enforcement of the DSA is relevant for those analytics whose execution is long lasting, because the user requesting the analytic execution could lose the related right while the analytics execution is still in progress. For instance, in the AT pilot, the DSA paired with some Data Bundles could require that the related data can be accessed and used only if the user is located within the airport premises. Hence, supposing that a user requests to execute a given analytic on a subset of those Data Bundles when he/she is located in the airport, the access is granted, and the analytic is started. However, if the user exits the airport when the analytic is still in progress, the DSA is violated, and the analytic execution must be interrupted (or suspended, waiting that the user enters the airport again).

The internal architecture of the ISI subsystem, which is shown in Figure 15, has been defined in the first year of the project, and it has not been modified in the second year of the project. For what concerns the internal components, in the second year of the project we focused on the design and implementation of the DMO Toolbox and of the Obligations Toolbox, as well as on the design and implementation of some DMOs and some Obligations that have been integrated in the framework through the previously mentioned component (as explained in details in Section 4.3 and Section 4.4). For what concern the IAI API, in this deliverable we give a very detailed description with respect to the description provided in deliverable D6.1. Instead, the other internal components, i.e., the Data Usage Control System, the Bundle Manager and the Buffer Manager have not been significantly modified or extended with respect to the version described in deliverable D6.1. However, for the sake of completeness, in this deliverable we report the description of these components as well which, in some points have been extended with more details with respect to the version provided in deliverable D6.1.

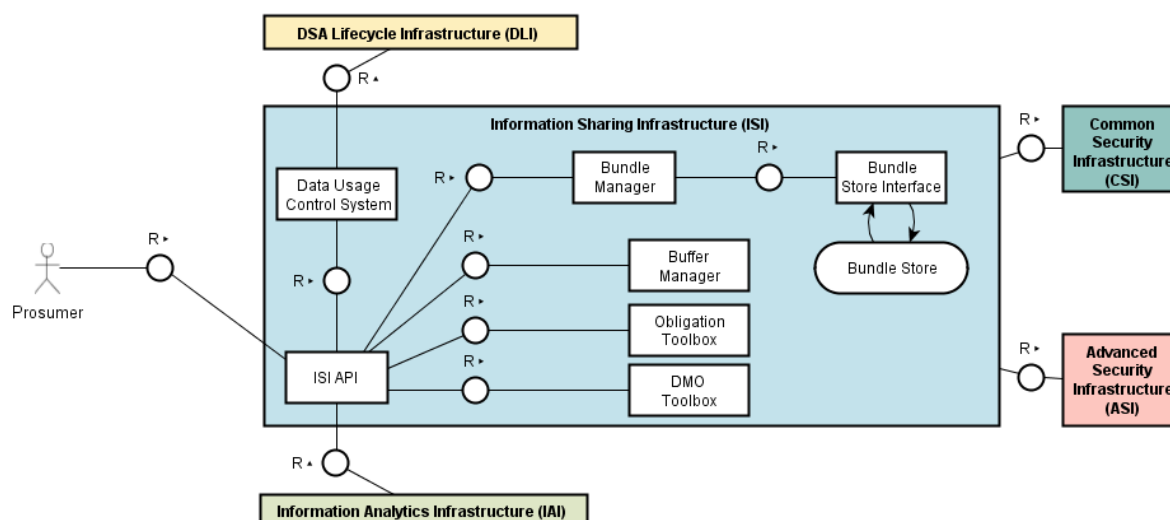


Figure 15 Information Sharing Infrastructure internal architecture.

The ISI subsystem is invoked by the prosumers of the E-CORRIDOR framework, directly (as shown by the Prosumer image on the left of Figure 15), or indirectly, i.e., by invoking the IAI subsystem to request the execution of an analytics. When prosumers invoke the ISI subsystem directly, they can use either a pilot specific graphical interface (which is customized for the specific type of data of the pilot) or the APIs provided by the ISI API component (described in Section 4.1). The prosumer invokes the ISI subsystem to perform the main operations on the data, i.e., CREATE, READ, and DELETE. Instead, for invoking the execution of analytics on the data, the prosumers invoke the ISI indirectly, because they interact with the IAI subsystem which, in turn, interacts with the ISI subsystem on their behalf to ask for the data required for the analytics execution. In particular, the IAI invokes the ISI exploiting the ISI API, invoking an operation called PREPARE DATA.

All the requests are received by the ISI API, which is the frontend of the ISI subsystem. For the execution of the READ and of the PREPARE DATA operations, the ISI API invokes the Buffer Manager (described in Section 4.8) to create a Virtual Data Lake (VDL) where the data whose usage is authorized by their DSAs will be stored. Then, the Bundle Manager (described in Section 4.5) invokes the Bundle Store (through the Bundle Store Interface) to retrieve all the Data Bundles matching the search criteria expressed in the request, extracts the related DSAs, and returns them to the ISI API which, in turn, invokes the Data Usage Control System to check them. If the Data Usage Control System authorizes the usage of some data, before sending it to the prosumer (in case of READ operation) or making them available on the Virtual Data Lake (in case of PREPARE DATA operation), the Data Usage Control System also determines the Obligations and the DMOs that must be performed before releasing them. For instance, the DSA could require that the IP addresses that are present in the data to be released must be anonymized. As last step, the ISI API, either returns to the prosumer the data (possibly elaborated by the DMO), or returns to the IAI API the link to the Virtual Data Lake (including the data possibly elaborated by the DMOs as well). This Virtual Data Lake is encrypted with a temporary key, to enhance data protection, and the temporary encryption key is sent back to the IAI API over a secured channel.

In case of CREATE operation, instead, both the data and the DSA reference are passed by the prosumer to the ISI API. The ISI components process the DSA in the same way as explained

above and, if the DSA authorizes the creation of a new Data Bundle, the DMOs dictated by the DSA are executed on the data, and the resulting data, along with the DSA, is used for the creation of a new Data Bundle by invoking the Bundle Manager.

The workflow of the DELETE operation is similar to the one of the READ one but, if the DSA is satisfied (i.e., there is a rule in the DSA which explicitly allows the cancellation of the Data Bundle), the Bundle Manager component is invoked to ask the cancellation of the Data Bundle from the Bundle Store.

4.1. ISI API

The ISI API is the frontend of the ISI subsystem, and it is invoked by the E-CORRIDOR users for executing operations on Data Bundle, such as CREATE, DELETE, SEARCH, READ, and MOVE Data Bundle. Moreover, the ISI API is also invoked by the IAI API (see Section 5.1) for preparing Virtual Data Lakes, i.e., set of data that will be used for the execution of analytics. The current version of the interfaces offered by the ISI API is shown in Figure 16.

ISI_API	
POST	<code>/isi-api/v1/dpo</code> Create a new DPO
GET	<code>/isi-api/v1/dpo/{dpo_id}/</code> Get an existing DPO content identified by its DPO_ID
DELETE	<code>/isi-api/v1/dpo/{dpo_id}/</code> Delete a DPO identified by DPO_ID
POST	<code>/isi-api/v1/dpo/receiveDPO</code> Moves a DPO -- part 2: receiving ISI
POST	<code>/isi-api/v1/move/dpo/{dpo_id}/</code> Moves a DPO -- part 1: sending ISI
POST	<code>/isi-api/v1/prepareData</code> Prepare a new VDL populated by a set of existing DPOs identified by DPO_IDs, the method returns a reference to the prepared data in a structure as defined by Buffer Manager
POST	<code>/isi-api/v1/releaseData/{sessionId}/</code> End a DPO Session identified by sessionId
POST	<code>/isi-api/v1/search/{store}/{longResultFlag}</code> Search on DPO Store (dpos) or DSA Store (dsas)

Figure 16 ISI REST API

The authentication of the user accessing the ISI API is performed using the Keycloak framework, which has been deployed as a part of the CSI subsystem, and it that has been integrated in the ISI for managing user authentication and single sign on.

From the technical point of view, the ISI API component provides a RESTful interface, which exposes the following methods:

- DPO (POST): this method implements the CREATE DATA BUNDLE operation, which creates a new Data Bundle starting from a data file, a DSA, and a set of

metadata. The parameters that the data producer should submit to the ISI API when invoking this method to create a new Data Bundle are:

- FILE-TO-SUBMIT: represents the file including the data that must be enclosed in the Data Bundle.
- INPUT-METADATA: is a JSON string which embeds all the metadata that are passed to the Data Bundle creation operation.

The metadata are organized as a list of attributes, and they are passed used the following format:

```
{
  "Request": {
    "Attribute": [
      {
        "AttributeId": ".....",
        "Value": ".....",
        "DataType": "....."
      },
      .....
      {
        "AttributeId": ".....",
        "Value": ".....",
        "DataType": "....."
      }
    ]
  }
}
```

Where the field `AttributeId` is used to specify the name of an attribute, the field `Value` is used to specify the value of such attribute, while the field `DataType` is used to specify the type of such attribute. Some example of attributes that can be passed in the Data Bundle creation request are:

- DSA-ID: the ID of the DSA that must be embedded in the Data Bundle that is being created to protect the data.
- RESOURCE-TYPE: the format of the data that are being embedded in the Data Bundle (e.g., txt, mp4, doc, etc.).
- RESOURCE-OWNER: the ID of the organization that will be the owner of the Data Bundle that is being created (e.g., CNR, HPE, etc.).
- START-DATE, START-TIME: the data in the Data Bundle that is being created refers to events which had place after the starting date and time.
- END-DATE, END-TIME: the data in the Data Bundle that is being created refers to events which had place before the end date and time.
- EVENT-TYPE: represents the type of event the data embedded in the Data Bundle refers to (e.g., log of a SSH service, video taken at gate 10).
- NORM: it is a Boolean flag that indicates whether the ISI must format the data in the Data Bundle using the STIX format [STIX].

Other attributes could be added in the future to satisfy raising needs of the pilots. As a matter of fact, since these attributes are passed in a single metadata parameter, this allows us to add any attribute we want to this string in the future without altering the API.

This method returns the following values:

- 201: the Data Bundle has been create successfully, and the ID of the Data Bundle is returned in the JSON payload as follows:

```
{
  "status": "SUCCESS",
  "content":
    {"additionalProperties": {"dpoId": "....."},
     ".....": "....."}
}
```

- 401: unauthorized: This means that the requestor was not authorized to perform this operation.
- 500: which means that an error occurred in the creation process. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

Figure 17 Graphical interface to interact with the E-CORRIDOR framework

Figure 17 shows an example of a graphical interface we developed to interact with the E-CORRIDOR framework. In particular, the screenshot shows the Upload page, which allows to create new Data Bundles. The user has to select the file including the data to be embedded in the Data Bundle, to insert a number of metadata (Event Type, Start Date, Start Time, End Date, EndTime, and Organization), and the ID of the DSA to be paired with the data. This graphical interface invokes the DPO API of the ISI with a POST call to create a new Data Bundle. This interface is generic, and it has been designed for testing purposes. We recall that each pilot use case will have its own graphical interface for interacting with the E-CORRIDOR framework, which will be customized according to the pilot's need.

- DPO (GET): this method implements the READ DATA BUNDLE operation, and it returns the raw data embedded in a Data Bundle. The parameter of this method are the following
 - DPO-ID: is the ID paired with the Data Bundle to be read, which is passed as suffix of the method URL.
 - INPUT-METADATA: is a string which embeds all the metadata that are passed to the Data Bundle creation operation. It is passed in the header of the method.

The metadata are organized as a list of attributes, using the same format shown for the Data Bundle creation operation. These metadata are relevant, for instance, for the evaluation of the DSA paired to the Data Bundle. Some examples of attributes that could be passed to this method are the following:

- ACTION-ID: is the unique ID of the analytics that will be executed on the data embedded in this Data Bundle. This field is used when the READ DATA

BUNDLE operation has been invoked as a consequence of a PREPARE DATA operation. In other words, this attribute is passed when the Data Bundle is read for being included in a VDL that will be used for executing an analytics. The analytics ID is necessary for the evaluation of the DSAs paired to the Data Bundle to determine whether the data can be read or not.

- SESSION-ID: this attribute is passed when the Data Bundle is read for being included in a VDL that will be used for executing an analytics. It represents the unique ID of the session created by the IAI for the execution of this analytics. The session ID is necessary because the Data Usage Control System stores it in its internal data structures and will use it to communicate back to the IAI API in case of DSA violation while the execution of the analytics is still in progress. The session ID will be used by the IAI API to identify which of the analytics that are in execution must be stopped as a consequence of the DSA violation.
- ACCESS-PURPOSE: this attribute specifies the purpose for which the Data Bundle is being read or the analytics has been requested. This attribute is collected when the read request or the analytics request is submitted, and it is relevant for the evaluation of the DSAs paired to the Data Bundle to determine whether the data can be read or not.

As stated also for the CREATE DATA BUNDLE operation, other attributes could be easily added in the future to satisfy raising needs of the pilots since these attributes are passed in a single metadata parameter formatted as a JSON string.

The READ DATA BUNDLE method returns following values:

- 200: the read operation has been executed successfully. The data is returned in the payload.
- 401: unauthorized: This means that the requestor was not authorized to read the data.
- 500: which means that an error occurred while reading the data paired in the requested Data Bundle. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

- DPO (DELETE): this method implements the DELETE DATA BUNDLE operation and it removes the Data Bundle whose ID is passed as input from the Bundle Store. The delete operation is executed only if it is allowed by the DSA paired with the Data. The parameters that are passed to this method are the following:
 - DPO-ID: is the ID paired with the Data Bundle to be deleted, which is passed as suffix of the method URL.
 - INPUT-METADATA: is a JSON string which embeds all the metadata that are passed to the Data Bundle creation operation.

The metadata are organized as a list of attributes, as shown for the Data Bundle creation operation.

This method returns following values:

- 200: the Data Bundle cancellation has been executed successfully.
- 401: unauthorized: This means that the requestor was not authorized to remove the requested Data Bundle from the Bundle Store.
- 500: which means that an error occurred when deleting the requested Data Bundle. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

- **SEARCH DATA BUNDLE:** this method allows the user to execute a query on the Bundle Store to search for a set of DPOs according to a given query. The parameter that the data consumer should submit to the ISI API when invoking this method are the following:
 - **STORE:** this parameter specifies whether the search operation must be executed on the Bundle Store or on the DSA Store. As a matter of fact, this operation can be used both for searching Data Bundles in the Bundle Store and also DSA in the DSA Store. This parameter is part of the method URL.
 - **LONG-RESULT-FLAG:** this flag determines whether the metadata have to be included in the list of DPOs returned by this method. This parameter is passed as suffix of the method URL.
 - **INPUT-METADATA:** is a JSON string which embeds all the metadata that are passed to the Data Bundle creation operation.

The metadata are organized as a list of attributes, as shown for the Data Bundle creation operation. The metadata that are currently available for the SEARCH DATA BUNDLE operation are:

- **SEARCH-STRING:** which defines the query. The search-string follows the JSON format, and it involves the metadata paired with the Data Bundles.

An example of search-string is the following:

```
{
  "combiningRule": "and",
  "criteria": [
    { "attribute": "event-type",
      "operator": "eq",
      "value": "gate10video" },
    { "attribute": "organization",
```

```

    "operator": "eq",
    "value": "CNR" } ]
}

```

The previous example uses the EVENT-TYPE metadata paired to the Data Bundles as search criteria. Further metadata could be added in the next version of the E-CORRIDOR framework if required by the pilot use case. The SEARCH DATA BUNDLE method returns following values:

- 200: the query has been executed successfully, and the list of DPO IDs is returned in the payload.
- 401: unauthorized: This means that the requestor was not authorized to perform this operation.
- 404: not found: No Data Bundles matching the search criteria have been found.
- 500: which means that an error occurred query execution. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```

{
  "error": "....."
}

```

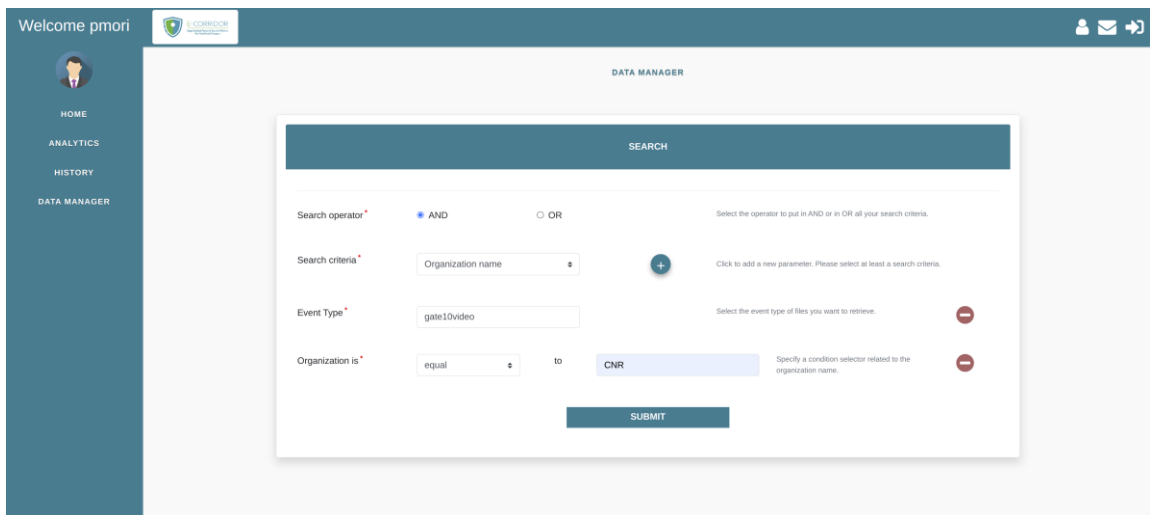


Figure 18 Graphical interface to interact with the E-CORRIDOR framework

Figure 18 shows an example of a graphical interface we developed to interact with the E-CORRIDOR framework. In particular, the screenshot shows the Search page, which allows to express the query to be executed on the metadata to select a number of Data Bundles. The query we defined in Figure 18 is the same we described before, i.e., we are looking for the Data Bundles having event type equal to “gate10video” and produced by organization CNR.

- **PREPARE DATA:** this method creates a Virtual Data Lake (VDL) including a set of raw data that will be used by an analytic. This method is invoked by the IAI API component as part of the workflow implementing the execution of an analytics. The parameter that must be passed to this method are the following:
 - **SERVICE-NAME:** is the unique ID of the analytics that will be executed on the data embedded in the VDL. The analytics ID is necessary for the evaluation of the DSAs paired to the Data Bundles that have been requested to be included in the VDL.
 - **INPUT-METADATA:** is a string which embeds all the metadata that are passed to the Data Bundle creation operation.

The metadata are organized as a list of attributes, exploiting the same format shown for the Data Bundle creation operation. Some examples of attributes that could be passed to in the PREPARE DATA request are the following:

- **DPO-IDS:** is the list of Data Bundle IDs that have been requested by the IAI API (i.e., by the consumer who invoked the IAI API) to be included in the VDL. The DSA of each of these Data Bundle is evaluated (see Section 4.2) to decide whether the related data will be part of the VDL or not, and also to determine whether some Data Manipulation Operations (see Section 4.3) must be executed on these data before copying them in the VDL.
- **SESSION-ID:** is the unique ID of the session created by the IAI for the execution of this analytics. The session ID is necessary because the Data Usage Control System will communicate it back to the IAI API in case of DSA violation while the execution of the analytics is still in progress. The session ID will be used by the IAI API to identify which of the analytics that are in execution must be stopped as a consequence of the DSA violation.
- **ACCESS-PURPOSE:** this attribute specifies the purpose for which the analytics has been requested. This attribute is collected when the analytics request is submitted, and it is relevant for the evaluation of the DSAs paired to the Data Bundles that have been requested to be included in the VDL.

Other attributes could be easily added in the future to satisfy raising needs of the pilots. The data in the VDL are encrypted with the symmetric scheme using a temporary encryption key. This method returns following values:

- **201:** the VDL has been created successfully. A JSON payload is returned to describe the VDL:

```
{
  "data": [
    {
      "DPO_id": ".....",
      "file": ".....",
      "code": ".....",
      "message": "....."
    }
  ],
  "datalake": {
```

```

    "URI": ".....",
    "message": "....."
    "datacipher": ".....",
    "datakey": ".....",
  }
}

```

This result is composed of two sections. The `data` section is an array which lists all the Data Bundles that have included in the VDL. For each of them, the field `DPO_id` reports the ID of the Data Bundle, the field `file` reports the name of the file in the VDL representing the data of such Data Bundle, and the fields `code` and `message` which report the outputs of the read operation for that Data Bundle. The `datalake` section, instead, reports the URI that must be used by the IAI to access the VDL, the encryption scheme that has been used for the temporary encryption of the files in the datalake, and the related decryption key.

- 401: unauthorized: This means that the requestor was not authorized to create the VDL.
- 500: which means that an error occurred when creating the VDL. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```

{
  "error": "....."
}

```

- **RELEASE-DATA:** this operation is meant to delete a VDL that has been prepared by the **PREPARE-DATA** operation and it is no longer needed. This operation is invoked by the IAI API when an analytics is terminated. The parameter that must be passed to this method is the following:
 - **SESSION-ID:** is the unique ID of the IAI session for which the VDL to be deleted was created. This parameter is passed as suffix of the invocation URL

This method returns following values:

- 200: the VDL cancellation has been executed successfully.
- 401: unauthorized: This means that the requestor was not authorized to remove the requested VDL.
- 500: which means that an error occurred when deleting the VDL. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```

{
  "error": "....."
}

```

```
}

```

- **MOVE DATA BUNDLE:** this method moves a Data Bundle from an ISI instance to another. The parameter that should be passed to this method are the following:
 - **DESTINATION:** is the address to the ISI instance where the Data Bundle will be moved.
 - **DPO-ID:** is the ID of the Data Bundle that will be moved from this instance of the ISI to the remote instance specified in the **DESTINATION** parameter. The **DPO-ID** is passed as suffix of the invocation URL.
 - **INPUT-METADATA:** is a string which embeds all the metadata that are passed to the method.

The metadata are organized as a list of attributes, as shown for the Data Bundle creation operation. In the case of the **MOVE DATA BUNDLE** operation, the relevant attributes are the same as the ones of the **READ DATA BUNDLE** operation.

This method returns following values:

- **200:** the **MOVE DATA BUNDLE** operation has been executed successfully.
- **401: unauthorized:** This means that the requestor was not authorized to move the requested Data Bundle.
- **500:** which means that an error occurred when moving the Data Bundle. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

- **RECEIVE DATA BUNDLE:** this method receives a Data Bundle that has been moved from another ISI instance. The parameter that should be passed to this method are the following:
 - **DPO-METADATA:** are the metadata that were paired to the Data Bundle on the remote ISI
 - **POLICY:** is the policy that was paired with the incoming Data Bundle, and that will be transformed in a DSA to be paired to the Data Bundle that is being created
 - **FILE-TO-SUBMIT:** represents the name of the file including the data that must be enclosed in the Data Bundle
 - **INPUT-METADATA:** is a string which embeds all the metadata that are passed to the method.

The metadata are organized as a list of attributes, as shown for the Data Bundle creation operation. In the case of the **RECEIVE DATA BUNDLE** operation, the

relevant attributes are the same as the ones of the CREATE DATA BUNDLE operation.

This method returns following values:

- 200: the RECEIVE DATA BUNDLE operation has been executed successfully, and the DPO ID is returned in the payload.
- 401: unauthorized: This means that the requestor was not authorized to receive the requested Data Bundle.
- 500: which means that an error occurred when creating the new Data Bundle. In this case, the method returns a JSON payload with the description of the error occurred, as follow:

```
{  
    "error": "....."  
}
```

4.2. *Data Usage Control System*

The **Data Usage Control System (DUCS)** is a component at the base of the E-CORRIDOR approach, because it is aimed at preserving the privacy of the data that are shared in the E-CORRIDOR framework for the execution of collaborative analytics. The functionalities and the internal architecture of this component have not undergone significant maturations or extensions compared to the version available at the end of the first year. However, some minor changes have been made. For instance, the Policy Information Points have been customized to manage the set of attribute requested by the pilots. Hence, for the sake of completeness, in the following we report an updated version of the description we gave in deliverable D6.1.

The DUCS protects the data that are shared by the data producers in the E-CORRIDOR framework following the Usage Control (UCON) model [PS2004]. From the implementation point of view, the DUCS is an extension of the **Usage Control System** presented in [CDLMM2016], which has been customized for operating data usage control. The Usage Control System processes distinct usage requests independently, while the DUCS is able to process a set of requests (asking the permission to access a number of distinct data pieces) at the same time, also allowing to enforce policies where the whole set of data pieces involved in the request is taken into account to decide whether one of these pieces of data can be used. In the following we will refer to these policies as “otherdata” policies.

The UCON model encompasses and extends the traditional access control models (such as Role Based Access Control, RBAC [SCFY1996], or Attribute Based Access Control, ABAC [HKFV2015]), by introducing two new decision factors besides **Authorizations: Obligations** and **Environmental Conditions**, as shown in Figure 19. Moreover, the UCON model also introduces the **continuity of the policy enforcement** to deal with changes in the access context.

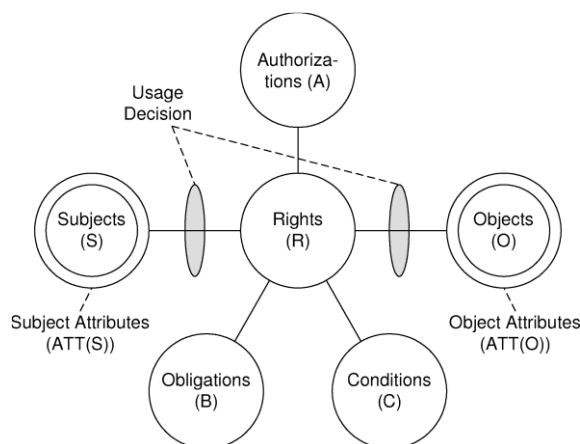


Figure 19 Usage Control Model [PS2004]

Since the proposed UCON model is an extension of the traditional ABAC access control model, Authorizations are expressed in the DSAs using rules defining constraints on the values of the attributes which characterize the subjects, i.e., the users of the E-CORRIDOR framework, and the objects, i.e., the data shared in the E-CORRIDOR framework. The DSA also allows to express constraints on the values of attributes representing the access context (e.g., date and time), thus implementing also UCON Environmental Conditions.

Obligations are actions that must be executed as a consequence of the access decision. For instance, sending a notification to an email address specified in the DSA is an example of obligation that must be executed by the DUCS. A specific kind of obligations are Data Manipulation Operations (DMOs), which are actions that are executed on the data embedded in the Data Bundle before being released for the execution of the analytic. For instance, deleting the last 3 digits of all the IP addresses included in a log file is a simple example of DMO that can be specified in the DSA and that must be executed on the data, i.e., the log file, before making it available for the analytic execution (see Section 4.3.2). Each specific pilot has its specific DMO that must be executed to ensure the required level of privacy of the shared data (as shown in Section 4.3).

The continuity of the policy enforcement is relevant especially in case of long-lasting computations (e.g., the execution of complex analytics), because the factors which initially granted the access to the data could change during the execution of the analytic in such a way that the access is not authorized anymore. For instance, a DSA could grant the access to a piece of data if the requesting subject is located inside a building, e.g., an airport in the AT pilot use case. Since the physical position of a person could change over time, when an access is in progress the DUCS should continuously check that the accessing subject is still located within the airport, and it should interrupt the access in case the subject exists the airport. For this reason, the access decision process is performed by the DUCS both when the access request is received, to decide whether the data can be used in the requested analytic (called *preAuthorization* in UCON), and also while the analytic is in progress (called *onAuthorization* in UCON), to check whether the rights to use the data is still valid or the analytic must be terminated.

When the execution of an analytic on a set of n Data Bundles is requested by a user, the DUCS receives an access request from the ISI API, which includes n XACML formatted access requests, one for each of the Data Bundles involved in the analytic execution. The DUCS processes all these requests and returns a response which, in turn, is a list of access

decisions (allowed/denied) stating whether each of the n Data Bundles can be used or not for the execution of the requested analytic. Moreover, for each of the Data Bundles that can be used, the DUCS response also specifies a set of DMO and Obligations that must be performed.

Since the access requests accepted by the DUCS are related to multiple Data Bundles, another relevant novelty introduced by the DUCS is the enforcement of rules which involves attributes of “otherdata”. As a matter of fact, in traditional authorization systems, the access control rules perform check on the values of the attributes of the subject requesting the access and on the data that is being accessed. The DSA policy language adopted in the E-CORRIDOR framework, in addition, allows policy makers to write rules which will be evaluated taking into account the other data that will be processed in the same request of the data the DSA refers to. For instance, this feature allows to write a policy which states that a piece of data cannot be used to execute an analytic which involves the data of a given competitor organization. Following the convention adopted in the DSA Editor, in the following, we will refer to these DSA rules as the “otherdata” rules.

The architecture of the DUCS, which is shown in Figure 20, exploits the Usage Control System, which is a UCON based general purpose authorization system able to manage single access requests, i.e., access requests concerning one generic resource only. The Usage Control System, in turn, extends the reference architecture of XACML based authorization systems, described in [XACML2013], for dealing with continuous policy enforcement and multiple access requests.

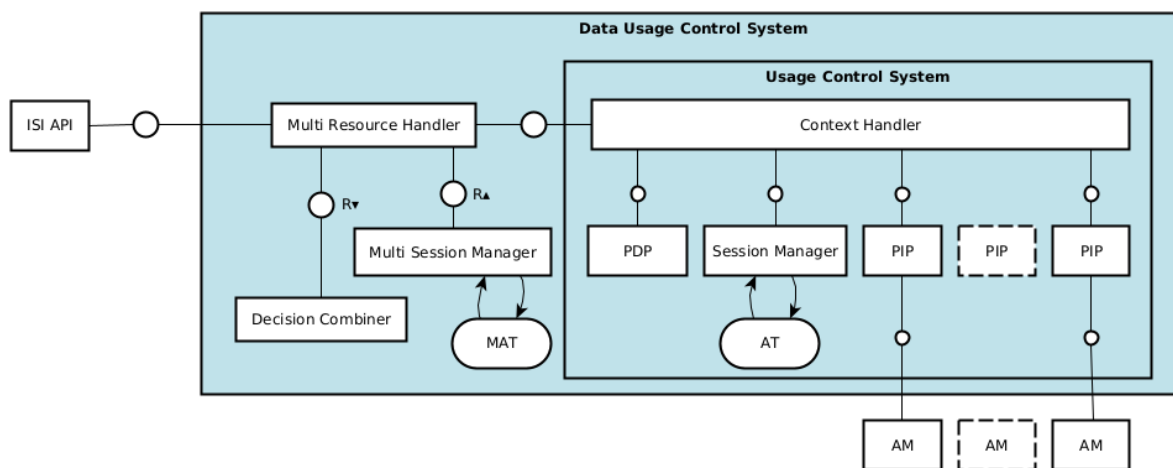


Figure 20 Data Usage Control System internal architecture

The **Multi-Resources Handler (MRH)** is the frontend of the DUCS and accepts the requests coming from the ISI API which, in turn, refer to requests to execute analytics on sets of Data Bundles, d_1, \dots, d_n . Each request received by the MRH includes a list of XACML formatted access requests, one for each of those Data Bundles, along with the related DSA (which, at this stage are expressed in enforceable format, i.e., in UPOL language [DLMMM2018]).

The MRH, at first, splits each of the received DSAs in two parts:

- the rules concerning the attributes of the data the DSA is paired with (data-DSA)
- the rules concerning the attributes of other data (otherdata-DSA).

Then, the MRH invokes the Context Handler (CH) component of the Usage Control System to evaluate the data-DSA over each request of the list (to determine whether the data in the related Data Bundle can be used for running the analytic). Section 4.2.1 will show how the CH will deal with each of these requests in order to return the access decision: allowed/denied. For each of the Data Bundles whose usage has been allowed, the MRH invokes the CH again in order to evaluate the other part of the DSA, the otherdata-DSA, on all the other Data Bundles whose usage has been admitted. This phase, for each Data Bundle d_i , produces the i^{th} line of the Compatibility Matrix (CM), where each position i,j specifies whether, according to its DSA, the Data Bundle d_i can be processed with the Data Bundle d_j . The matrix will be passed by the MRH to the Decision Combiner (DC) component which will select the final set of Data Bundle that can be used to perform the requested analytics.

The **Multi Session Manager (MSM)** component extends the functionality of the Session Manager (SM) component of the Usage Control System to manage multi resource access requests. The main task of this component is to keep track of a set of data involved in each request. To this aim, this component exploits a MultiAccess Table (MAT) to store the meta-data regarding these multi resource access requests. In particular, each entry in the MAT represents a session created for a multi resource access request, and includes the list of the sessions created in the SM for each of the Data Bundle involved in such multi resource access request.

The **Decision Combiner (DC)** component is invoked by the MRH after that all the decision processes concerning the Data Bundles involved in the request have been completed, to determine the final set of Data Bundles on which the requested analytics will be performed. In particular, as previously explained, each Data Bundle d is paired with a DSA which includes rules defining constraints concerning the set of other Data Bundles with which d can be processed. For instance, a DSA could state that the Data Bundle d cannot be exploited by the analytics a if the other Data Bundles processed with d do not satisfy a set of conditions defined on the attributes paired with such Data Bundles. The DC component takes as input the Compatibility Matrix computed by the MRH and determines the set of Data Bundles on which the requested analytic can be executed in such a way that both a specified objective function and the policies paired with all the Data Bundles included in such set are satisfied. Distinct objective functions can be defined for each E-CORRIDOR pilot, depending on the requirement of the specific analytics to be performed. For instance, an objective function could be the maximization of the number of Data Bundles involved in the analytics.

4.2.1 Usage Control System

The Usage Control System is aimed at regulating the usage of a (single) resource (data or service) following the UCON model. This system is invoked multiple times by the MRH to implement the data usage control.

The main components of the Usage Control Systems are the following.

The **Context Handler (CH)** component is aimed at coordinating the process of evaluation of single access requests, i.e., the evaluation of the access request to a Data Bundle against the related DSA. The CH is invoked by the MRH, and it invokes the other components of the Usage Control System:

- Policy Information Points for attribute retrieval;
- Policy Decision Point for DSA evaluation;
- Session Manager for session creation.

To execute the attribute retrieval step, the CH invokes all the Policy Information Points which will take care of the actual retrieval process, as shown in the following of this section. Once all the required attribute values have been retrieved (and embedded in the access request), the CH invokes the Policy Decision Point for the DSA evaluation exploiting the collected attribute values. If the response returned by the PDP is positive, i.e., the DSA allows the usage of the Data Bundle for executing the requested analytic, the CH will interact with the Session Manager (SM) to create a new session which will represent the ongoing usage of the related Data Bundle.

The **Attribute Managers (AMs)** are services providing the current values of the attributes that are used in the DSAs. These services are not necessarily deployed with the E-CORRIDOR framework, and they are dependent on the specific security and privacy needs of the pilots. Each pilot use case could decide to exploit services that already exist as AMs, and/or to deploy its own custom services to manage the specific attributes that are needed in its scenario. For instance, a pilot could exploit an existing Identity Provider (IdP), for instance the Keycloak service that we deployed as a part of the CSI subsystem, as AM to provide some attributes of the users such as the company the user belongs to. Alternatively, new services can be deployed by a company, for instance to manage an attribute representing the role of the in the company. AMs provide their custom interfaces to allow external clients to retrieve the current values of the attributes they manage, and the Policy Information Points exploit these interfaces to interact with them. Some of them could also provide an interface to update attribute values. At the end of the second year of the project the pilots are using for their DSAs the attributes listed in Table 4.

Table 4 Attributes defined by the Pilots

Attribute				Pilot		
Category	Name	Term in the DSA policy	Description	AT	S2C	ISAC
Subject	Role	hasRole	Role of the user in the Organization	x	x	x
Subject	Organization	hasOrganization	Name of the Organization the user belongs to	x	x	x
Subject	Nationality	hasCountry	Name of the Country the user belongs to	x		x
Subject	Physical position	hasPhysicalPosition	Physical position of the user (specified as a string, e.g., gate 10, entrance, parking)	x	x	
Subject	Sector	hasTransportationSector	Transportation sector the user belongs to (e.g., railway, aviation,	x	x	x

			automotive)			
Subject	Group	isMemberOf	Name(s) of the Group(s) the user belongs to			
Subject	Owner	isOwner	The IDs of the Data Bundle of which the user is the owner		x	
Data	Producer	hasProducer	Name of the Organization who causes the data to be produced	x		x
Data	ProducerAppliance	hasProducerAppliance	Kind of the service or device which produced the data	x		
Data	ProducerApplianceOwner	hasProducerApplianceOwner	Owner of the service or device which produced the data	x		x
Data	ProductionPhysicalPosition	hasProductionPhysicalPosition	Physical position of the device which produced data when the data were produced	x		x
Data	Type	hasType	Type of the data	x		x
Data	StartValidity	hasStartTime	The data are valid after this date. In this way, the data produce can invalidate data, thus preventing previously allowed users to access it.	x		
Data	EndValidity	hasEndTime	The data are not valid after this date. In this way, the data produce can invalidate data, thus preventing previously allowed users to access it.	x		

The **Policy Information Points (PIPs)** are the interfaces exploited by the CH for interacting with the Attribute Managers to retrieve the current values of the attributes required by the PDP to evaluate the DSA. Since Attribute Managers typically provide different protocols, as previously explained, the PIPs are the adapters to communicate with these AMs because the CH is unaware of those specific protocols. In particular, the proposed architecture includes a chain of PIPs which provide the same interface to the CH (attribute *retrieve*,

subscribe/unsubscribe and *update*), while each PIP implements the specific protocol to interact with the Attribute Manager is paired with, and the specific algorithm to perform the requested operation and to provide the required information. The *retrieve* interface is invoked by the CH to get the updated values of the attributes managed by the PIP. In particular, the PIP embeds the collected values in the original access request. Besides collecting the current values of attributes, PIPs are also in charge of monitoring the values of such attributes to detect when an update occurs. In particular, when the CH invokes the *subscribe* interface of a PIP, this PIP starts monitoring the value of the corresponding attribute, and it notifies the CH as soon as this value changes. As a matter of fact, since this change could lead to a violation of the DSAs of the data bundles that are being accessed, once the CH receives an attribute update notification, it must trigger the re-evaluation of such DSAs and, in case of violation, it must interrupt the ongoing accesses. The *unsubscribe* interface is used to stop the attribute monitoring, e.g., when the related access has been interrupted. Finally, the *update* interface is used to change the current value of the attribute. In order to deal with the attributes proposed by the Pilots (and listed in Table 4), the following PIPs have been defined and integrated:

- LDAP PIP, in charge of retrieving user attributes from an LDAP service. This PIP interacts with the LDAP service managing some of the attributes of the E-CORRIDOR users;
- MySQL PIP, in charge of retrieving user attributes from a MYSQL service. This PIP interacts with the LDAP service managing some of the attributes of the E-CORRIDOR users;
- Metadata PIP, in charge of retrieving data attributes from the metadata paired with Data Bundles;
- DLI PIP, which interacts with the DLI subsystem. This PIP is integrated by default in the Usage Control System, and it is in charge of retrieving the current status and version of the DSA under evaluation. This is to ensure the validity of the DSA and to deny access to data if necessary;
- Keycloak PIP, which interacts with the Keycloak service which has been integrated in the E-CORRIDOR framework for managing user authentication and single sign on. We plan to move most of the user attributes on this AM by the end of the project.

The **Policy Decision Point (PDP)** evaluates the access requests against the DSAs and produces the access decision (Permit/Denied). As previously explained, before being submitted to the PDP, the access request is enriched by the CH with the current values of the attributes collected by the PIPs from the AMs. The access request is expressed in XACML format. Moreover, before submitting the DSA (which is written in UPOL language) to the PDP for the evaluation, the MRH properly elaborates it properly converting the UPOL policy to a XACML compliant policy. Hence, the PDP is implemented by using a standard XACML engine, such as WSO2 Balana [BAL2021].

The **Session Manager (SM)** is the component of the Usage Control System in charge of keeping track of the ongoing usage sessions, i.e., of the accesses that have been permitted and that are currently in progress. It exploits an Access Table, which is implemented using a MySQL Data Base, to store the meta-data regarding these ongoing sessions. Each entry in the Access Table refers to an ongoing access to a Data Bundle, and it includes the session-id, the

access request, and the DSA in UPOL format. A new entry is created in the Access Table every time a new access request is allowed by the PDP because of the DSA evaluation, and this entry is deleted when the related access is terminated. When a PIP detects that the value of one of the attributes that are currently monitored has changed, the SM component is queried to obtain the list of the ongoing accesses that could be affected by this event, because the DSA paired to Data Bundle that is being accessed includes (at least) a rule (prohibition, authorization or obligation) involving such attribute.

4.3. *Data Manipulation Operations Toolbox*

The Data Manipulation Operations toolbox is the component of the ISI subsystem in charge of the execution of Data Manipulation Operations (DMO) on the data when required by the DSAs. DMOs are operations that are executed by the E-CORRIDOR framework on the data embedded in the Data Bundle before copying them on the Virtual Data Lake, thus making them available to the IAI subsystem for the execution of the analytics, or to the users for downloading. The idea of DMOs is to provide to data producers more control on the data they share. In fact, the DMO are executed on the data for altering them in order to modify or remove specific part of them (for instance, anonymizing IP addresses or email addresses). The DSA could require that different DMOs are executed on the same raw data depending analytic to be executed on such data. The DSA could also require that multiple (i.e., a sequence of) DMOs must be executed to prepare the data for a specific analytic.

The sequence of DMOs that must be executed on each piece of data is determined by the DUCS when it evaluates the DSA for all the Data Bundle involved in a request. The DUCS returns to the ISI API a response which includes, for each Data Bundle, the list of DMO to be executed on the related data, and the ISI API invokes the DMO Toolbox to execute each of these operations before making the data available to the requesting user or to copy them in the Virtual Data Lake for the execution of an analytic.

Since each pilot has its specific data format and its specific privacy requirements on such type of data, which could also depend on the analytics defined by the pilot on the data, the DMO Toolbox has been designed to be able to easily integrate distinct DMOs that work on distinct data types. DMOs can be developed either by the pilots or by the technology providers of the E-CORRIDOR project by following a *plugin* approach, thus easing the addition of new DMOs to the E-CORRIDOR framework even when it has been already deployed. This approach also protects the source code developed by the technology providers.

Hence, the Data Manipulation Operation Toolbox works as a kind of frontend for the set of DMOs integrated in the ISI subsystem. To this aim, this component exposes a unique API, `executeDMO`, and, depending on the value of the invocation parameters, it invokes the right DMO.

In order to be integrated in the E-CORRIDOR framework through the DMO Toolbox, DMOs must be implemented as `https` RESTful services exposing their interface by following the OpenAPI specification [OpenA2020], and providing the endpoints described in the following. The RESTful services implementing DMOs should be available into a docker container image, which is typically (but not necessarily) deployed on the same machine where the E-CORRIDOR framework has been deployed.

Figure 21 shows the internal architecture of the DMO Toolbox component. On the right side of the image we can see the set of DMO Services that have been developed by third parties,

which are RESTful services following the protocol we defined. The DMO toolbox has an internal registry, the DMO Registry, where each entry represents a registered DMO Service and reports the unique name of the DMO and the URL to invoke the service. This registry is populated by reading the `dmo.yaml` configuration file.

Finally, the DMO Frontend is the subcomponent which is invoked by the ISI API to invoke the `executeDMO`, and that reads the DMO Registry for finding the URL to be invoked, and that actually invokes the DMO Service.

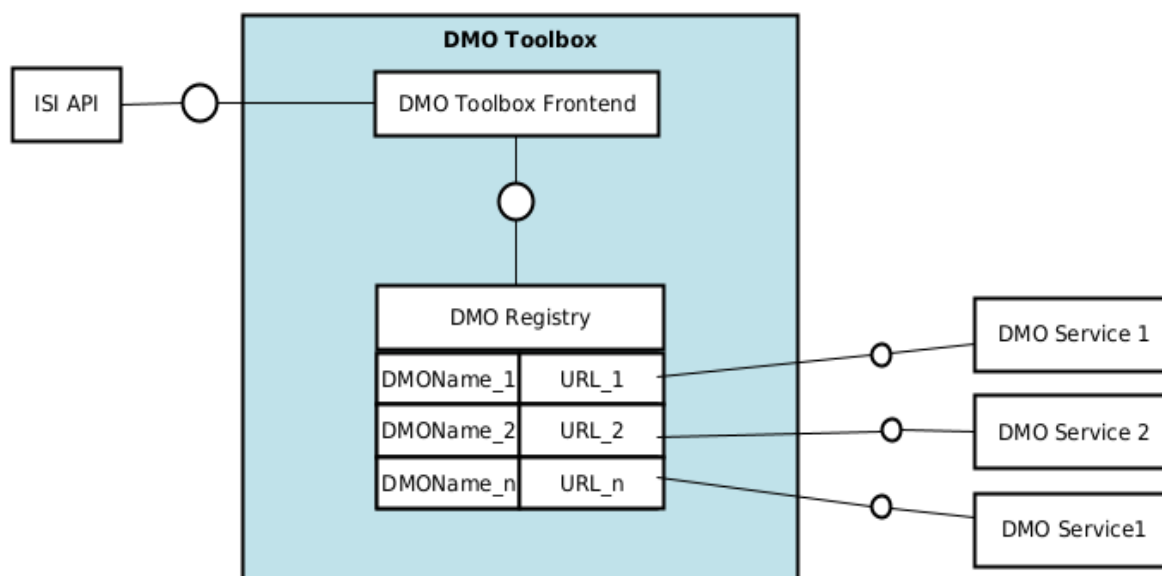


Figure 21 DMO Toolbox internal architecture

Each DMO Service must provide the following endpoint:

https://0.0.0.0/dmo_process

The **dmo_process** endpoint is invoked by the DMO Toolbox choosing the proper address of the DMO service to invoke. This endpoint is invoked through a HTTP POST method, providing the data required to run the DMO by passing the following JSON payload:

```

{
  "dmo_file": ".....",
  "dmo_params": "....."
}
  
```

The **dmo_file** field reports the unique name of the file on the temporary datalake on which the DMO must be executed, while the field **dmo_params** is used to send to the DMO some

parameter which are specific of such DMO, and they are specified in the DSA where the DMO has been invoked. The DMO Toolbox is agnostic with respect to these parameters, whose semantic is known only by the specific DMO service. The interaction protocol between the DMO Toolbox and the DMO service is synchronous. The result of the DMO is the modified file on the datalake, and the DMO service simply sends back a status code as response when the execution is finished and the file is ready to be used:

- 204: which means that the DMO has been executed successfully, and the data resulting from the execution of the DMO has been stored in the file specified in the **dmo_file** field of the request, substituting the original data.
- 500: which means that an error occurred. In this case, the DMO service must return a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

In the following we give a detailed description of the DMOs that have been currently implemented and deployed in the E-CORRIDOR framework:

4.3.1. Face Anonymization

Requirements in the AT pilot have expressed the need for analysing videos recorded by the environmental cameras deployed in the airport terminal and in the train station. Data collected include the faces of all the passengers passing through the public spaces of the transportation premises. Since the analytics performed by these cameras may not need to identify the passenger, a facial reduction module has been integrated in the DMO toolbox. The bottom-line idea is to give data consumers the ability to access the video content, while preserving the privacy of the passengers by blurring or blocking out the faces in a video (according to the DSA expressed by the data producer).

Working principle: The face anonymization DMO works in a few phases. Given a video in input it first detects the people in the scene, then the faces are detected and blurred. The bounding boxes (imaginary rectangles used as point of reference for object detection) surrounding the people are tracked to keep the face blurred while the people move. The diagram in Figure 22 represents at high level the described process.

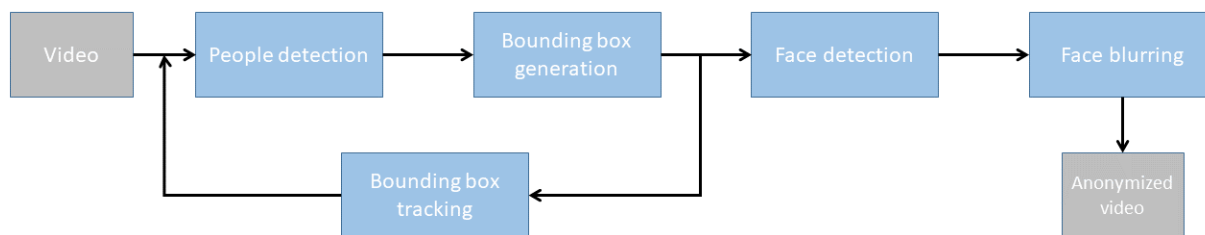


Figure 22 Face anonymization pipeline

The described process turns out to be computationally complex with an accuracy that often do not reach 100%. This is due to the accumulation of errors in the different activities performed in the pipeline. In particular, the process requires to predict the direction of the moving objects.

If the process is performed offline with a pre-recorded video, it is possible to obtain a higher quality of the process by paying more in computational terms. In such a case, the tracking is not performed, and each image is processed independently. In the complexity analysis, it is however worth to consider that modern IP security cameras offer a frame rate ranging from 30 FPS (frames per second) to 60 FPS or more, and a resolution from a few to tens of MP (mega pixels). To improve the performance, it is therefore advisable to adopt some optimization techniques derived from video and image compression in such cases.

Design and Exposed REST API: The face anonymization DMO has been developed in Python. Among the other required imports, the component exploits the OpenCV (an open source and real-time computer vision library) and TensorFlow (an open-source library for performing training and inference in machine learning and artificial intelligence applications) libraries.

The face anonymization DMO is fully integrated in the DMO toolbox. When the the face anonymization DMO is invoked through the `dmo_process` interface, through the `dmo_params` field of the JSON payload it is possible to provide options for tuning the process e.g., by resizing the output file at a given resolution (`size`: in pixel), compressing the file (`output_format`: MJPEG or XVID), or reducing the framerate (`jump_frame`: as division from the frame per seconds in the original video) to increase analytics performance.

Implementation and Integration Status at M24: At the time of writing a first implementation has been completed and the component is fully integrated in the DMO toolbox. It is therefore possible to specify DSA policy that invoke the face anonymization in the video by requesting the system to *AnonymizeFaces*. An example of its use can be found in the obligation clause of the DSA policy reported in Section 3.1.1.

Future maturation efforts will be oriented at improving the performance of the component. Also, if in any of the use cases presented in the pilots the needs for customizing the blurring will be raised some blur modes/options may be exposed.

Results: In the AT pilot, the face anonymization is applied on the videos recorded by the environmental cameras. Therefore, the covered scene usually encompasses the presence of a high number of passengers moving relatively fast (e.g., because they are in a hurry to get to

their gate). An example of the output video generated by the face anonymization DMO is reported in the screenshot of Figure 23.

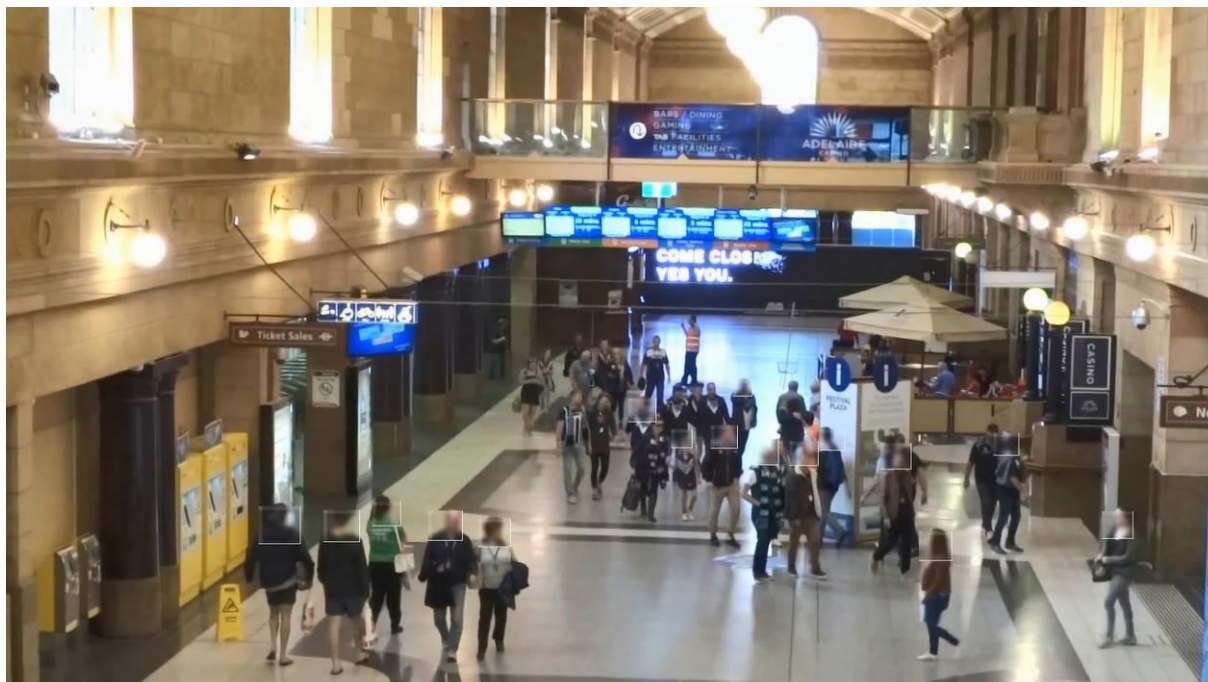


Figure 23 Facial reduction DMO

4.3.2. IP Addresses Anonymization

Since analysis performed on the shared data may not require personally identifiable information, including IP addresses, a data anonymization module has been integrated into the DMO toolbox. The main goal of this integration is to allow data consumers to access and use that data and, at the same time, keep sensitive data secret. With the combination of the DSA attached to the dataset, the data anonymization module can mask IP addresses.

Working principle: The data anonymization module uses the original data version to analyze it and identify sensitive information, including specified IP addresses. Then, the module masks each IP address depending on the data anonymization module configuration, i.e., completely or partially masking IP address. Figure 24 describes the IP anonymization process at a high level.

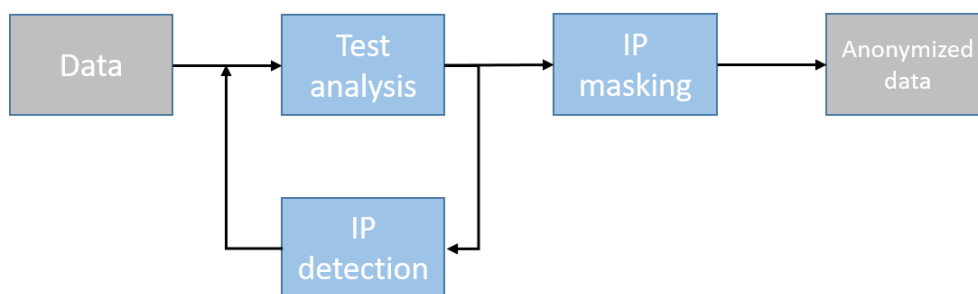


Figure 24 IP Anonymization process

The IP anonymization process can be invoked before performing analytics on the data or before releasing data to the consumer. Hence, according to the DSA paired to the data, only particular consumers can access the original version of data, while the anonymized version of that data will be available for a more broad audience if other conditions specified in the DSA are met.

Design and Exposed REST API: The data anonymization module has been implemented in Python. It uses the combination of multiple open-source advanced Natural Language Process (NLP) libraries, including spaCy and NLTK, to support extensive sensitive data anonymization. Apart from the IP masking, the data anonymization module can also work with other types of sensitive data, including names, organization names, date of birth, phone numbers, email addresses, etc.

Implementation and Integration Status at M24: The first version of the data anonymization module has been implemented and fully integrated into the DMO toolbox. The data producers can define DSA and request the IP anonymization on the corresponding data by specifying *AnonymizeIPByMasking* or *AnonymizeIPByRandomization* during the DSA definition process.

Future maturation efforts will be focused on the integration of other data anonymization module functionalities. Moreover, the module may be easily configured to address additional potential needs of the pilots.

```

ID,ts_year,ts_month,ts_day,ts_hour,ts_min,ts_second,te_year,te_month,te_day
,te_hour,te_min,te_second,td,sa,da,sp,dp,pr,_flag1,_flag2,_flag3,_flag4,_fl
ag5,_flag6,fwd,stos,ipkt,ibyt,opkt,obyte,_in,out,sas,das,smk,dmk,dtos,_dir,n
h,nhb,svln,dvln,ismc,odmc,idmc,osmc,mpls1,mpls2,mpls3,mpls4,mpls5,mpls6,mpl
s7,mpls8,mpls9,mpls10,cl,sl,al,ra,eng,exid,tr,icmp_dst_ip_b,icmp_src_ip,udp
_dst_p,tcp_f_s,tcp_f_n_a,tcp_f_n_f,tcp_f_n_r,tcp_f_n_p,tcp_f_n_u,tcp_dst_p,
tcp_src_dst_f_s,tcp_src_tftp,tcp_src_kerb,tcp_src_rpc,tcp_dst_p_src,smtp_ds
t,udp_p_r_range,p_range_dst,udp_src_p_0,attack_t,attack_a

1,2020,1,25,3,44,57,2020,1,25,3,44,57,0.28,193.219.74.172,83.169.5.121,5975
8,3306,TCP,.,A,P,.,.,F,0,8,10,1080,0,0,762,556,2847,8972,0,0,0,0,0.0.0.0,0.
0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00
:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,1970-01-01
03:00:00.000,normal,normal,normal,normal,normal,normal,normal,normal,normal
,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,none
,0
  
```

```

2,2020,1,25,3,44,52,2020,1,25,3,44,57,4.73,184.73.156.247,193.219.75.85,443
,58895,TCP,,A,,,,,F,0,0,10,7300,0,0,707,588,14618,2847,0,0,0,0,0.0.0.0,0
.0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:0
0:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,1970-01-01
03:00:00.000,normal,normal,normal,normal,normal,normal,normal,normal,normal
,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,none
,0
3,2020,1,25,3,44,57,2020,1,25,3,44,57,0,83.171.40.3,80.82.77.132,6666,56667
,TCP,,A,,R,,,,,0,0,5,200,0,0,922,556,2847,202425,0,0,0,0,0.0.0.0,0.0.0.0,
0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00
,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,1970-01-01
03:00:00.000,normal,normal,normal,normal,normal,normal,normal,normal,normal
,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,none
,0
4,2020,1,25,3,44,38,2020,1,25,3,44,57,19.29,95.108.213.17,158.129.192.240,6
2443,80,TCP,,A,,,,,S,F,0,164,20,1080,0,0,556,558,13238,2847,0,0,0,0,0.0.0.
0,0.0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:0
0:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,1970-01-01
03:00:00.000,normal,normal,normal,normal,normal,normal,normal,normal,normal
,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,none
,0
.....

```

Figure 25 Excerpt of a netflow output file

```

ID,ts_year,ts_month,ts_day,ts_hour,ts_min,ts_second,te_year,te_month,te_day
,te_hour,te_min,te_second,td,sa,da,sp,dp,pr,_flag1,_flag2,_flag3,_flag4,_fl
ag5,_flag6,fwd,stos,ipkt,ibyt,opkt,obyt,_in,out,sas,das,smk,dmk,dtos,_dir,n
h,nhb,svln,dvln,ismc,odmc,idmc,osmc,mpls1,mpls2,mpls3,mpls4,mpls5,mpls6,mpl
s7,mpls8,mpls9,mpls10,cl,sl,al,ra,eng,exid,tr,icmp_dst_ip_b,icmp_src_ip,udp
_dst_p,tcp_f_s,tcp_f_n_a,tcp_f_n_f,tcp_f_n_r,tcp_f_n_p,tcp_f_n_u,tcp_dst_p,
tcp_src_dst_f_s,tcp_src_tftp,tcp_src_kerb,tcp_src_rpc,tcp_dst_p_src,smtp_ds
t,udp_p_r_range,p_range_dst,udp_src_p_0,attack_t,attack_a
1,2020,1,25,3,44,57,2020,1,25,3,44,57,0.28,193.219.74.172,83.169.000.000,59
758,3306,TCP,,A,P,,,,,F,0,8,10,1080,0,0,762,556,2847,8972,0,0,0,0,0.0.0.0,
0.0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:
00:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,00:00.0,normal,normal,normal,normal,normal,normal,nor
mal,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,n
ormal,normal,none,0
2,2020,1,25,3,44,52,2020,1,25,3,44,57,4.73,184.73.156.247,193.219.000.000,4
43,58895,TCP,,A,,,,,F,0,0,10,7300,0,0,707,588,14618,2847,0,0,0,0,0.0.0.0
,0.0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00
:00:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,00:00.0,normal,normal,normal,normal,normal,normal,normal,
normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,n
ormal,normal,none,0
3,2020,1,25,3,44,57,2020,1,25,3,44,57,0,83.171.40.3,80.82.000.000,6666,5666
7,TCP,,A,,R,,,,,0,0,5,200,0,0,922,556,2847,202425,0,0,0,0,0.0.0.0,0.0.0.0
,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00
,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,00:00.0,normal,normal,normal,normal,normal,normal,normal,
normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,n
ormal,normal,none,0
4,2020,1,25,3,44,38,2020,1,25,3,44,57,19.29,95.108.213.17,158.129.000.000,6
2443,80,TCP,,A,,,,,S,F,0,164,20,1080,0,0,556,558,13238,2847,0,0,0,0,0.0.0.

```

```

0,0.0.0.0,0,0,00:00:00:00:00:00,00:00:00:00:00:00,00:00:00:00:00:00,00:00:0
0:00:00:00,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-0,0-0-
0,0,0,0,0.0.0.0,0/0,1,00:00.0,normal,normal,normal,normal,normal,normal,nor
mal,normal,normal,normal,normal,normal,normal,normal,normal,normal,normal,n
ormal,normal,none,0
.....

```

Figure 26 Excerpt of a netflow output file after the execution of the IP address anonymization by IP Masking DMO

Figure 26 shows the result of the application of the IP address anonymization by IP Masking DMO on the file shown in Figure 25, which is produced by the Netflow application which is typically used on some router devices to monitor the network traffic. We can notice that in Figure 26 the last 2 numbers of the source IP addresses have been cancelled (i.e., substituted with the two numbers: 000.000).

4.3.3. Passenger Face Extraction

According to the AT pilot requirements, videos recorded and images captured by airport/train station kiosk cameras need to be analysed. In the collected videos and images, multiple passenger faces can be captured in addition to the main passenger face in front of the kiosk. In order to extract only the main passenger face, a DMO exploiting recorded videos and captured images using a depth camera is required.

Working principle: The DMO for passenger face extraction consists of few steps. Given an RGB-D video with depth information and/or an image with depth information recorded/captured by using a depth camera, e.g., Intel RealSense D435 camera, in input, the DMO first extracts the number of passengers' faces in the scene then builds a 3D pointcloud. The latter is of paramount importance to identify the most salient passenger, i.e., the main passenger, by using the depth information given that the main passenger's distance from the airport/train station kiosk is less than the distance of the background passenger faces.

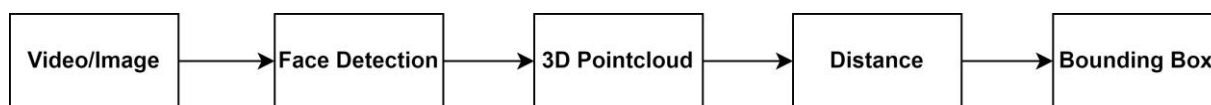


Figure 27 Passenger face extraction pipeline

In the face detection step, the DMO allows to locate the passenger faces in the scene. 3D pointcloud allows to build a 3D environment based on the video/image depth information collected by the depth camera. In distance step, measurement between the airport/train station and the passenger faces are calculated. A bounding box is assigned to the nearest passenger face of the kiosk.

The development of this DMO is currently in progress.

4.4. Obligations Toolbox

The Obligations toolbox is the component of the ISI subsystem in charge of the execution of Obligations. Obligations are actions that must be executed by the DUCS as a consequence to an access to a Data Bundle but they do not involve the data embedded in the Data Bundle. The internal design of the Obligations Toolbox component is the same as the one of the Data Manipulation Toolbox. In the same way as DMOs, Obligations are meant to provide to data producers more control on the data they share. In fact, the Obligation triggers the execution of operations, such as sending a notification to the data producer, when the data are used. The execution of Obligation is determined by the DSA, which could require that different Obligations are executed on the same data when different analytic are executed on such data. The DSA could also require that multiple (i.e., a sequence of) Obligations must be executed to prepare the data for a specific analytic.

As for the DMOs, by the DUCS determines the sequence of Obligations that must be executed for each piece of data when it evaluates the DSA for all the Data Bundles involved in a request. The DUCS returns to the ISI API a response which includes, for each Data Bundle, the list of Obligations to be executed, and the ISI API invokes the Obligations Toolbox to execute each of these operations.

Similarly to the DMO Toolbox, to accommodate the specific needs of each pilot, the Obligation Toolbox has been designed to easily integrate several and distinct Obligations, that can be developed either by the pilots or by the technology providers of the E-CORRIDOR framework by following a *plugin* approach. In this way, new obligations can be added to the E-CORRIDOR framework even when it has been already deployed in case of raising needs from the pilots.

In particular, in order to be integrated in the E-CORRIDOR framework through the Obligation Toolbox, such Obligations must be implemented as https RESTful services exposing their interface by following the OpenAPI specification [OpenA2020]. Similarly to the DMO, such RESTful services should be available into a docker container image, which is typically (but not necessarily) deployed on the same machine where the E-CORRIDOR framework has been deployed. The internal architecture of the Obligation Toolbox is identical to the one of the DMO Toolbox shown in Figure 21, and the JSON messages exchanged between the Obligation Toolbox and the Obligation services are the same as the ones exchanged among the DMO Toolbox and the DMO services.

Hence, the Obligation Toolbox is a frontend for the set of Obligations integrated in the ISI subsystem, it exposes a unique API, `executeObligation`, and, depending on the value of the invocation parameters, it invokes the right Obligation Service, which runs in its own container.

At the end of the second year of the project we implemented and deployed the Notification Obligation, described in the following.

4.4.1. Notifications

Since prosumers may need to receive a message about a new operation executed on their data, the E-CORRIDOR framework also integrates an obligation implementing a set of notification functionalities. Depending on the contact type (e.g., SMS or e-mail) defined in the DSA for the dataset, the notification engine may notify stakeholders or data owners about the operation requested on that dataset. For this reason, data prosumers can specify in an obligation rule of the DSA the `NotifyUserOn` obligation. The `NotifyUserOn` operation allows data producers to

specify contact details of the stakeholders that should receive notification from the system. The notification message includes information about the DPO uploaded to the system, including DPO-ID and the corresponding URL. Data consumers can use that URL to access the dataset. The system will evaluate the access request from the data consumer by comparing attributes of the requestor, DPO, and context information against security policies specified in the corresponding DSA. The E-CORRIDOR framework may send notification messages right after data producers upload their dataset to the system or when data consumers requests access to that dataset.

4.5. *Bundle Manager*

The Bundle Manager is in charge of performing two different operations. During the packaging operation, the Bundle Manager creates a new Data Bundle by pairing the selected DSA with the provided data and extending this bundle with relevant metadata. Instead, when a user requests to read a Data Bundle or to use it for the execution of an analytics, the Bundle Manager is in charge of retrieving and decrypting it, and to extract the DSA to be processed by the DUCS (see Section 4.2).

This component has not undergone significant maturations or extensions compared to the version available at the end of the first year. Hence, for the sake of completeness, in the following we report the description we gave in deliverable D6.1.

Figure 28 illustrates the Data Bundle structure.

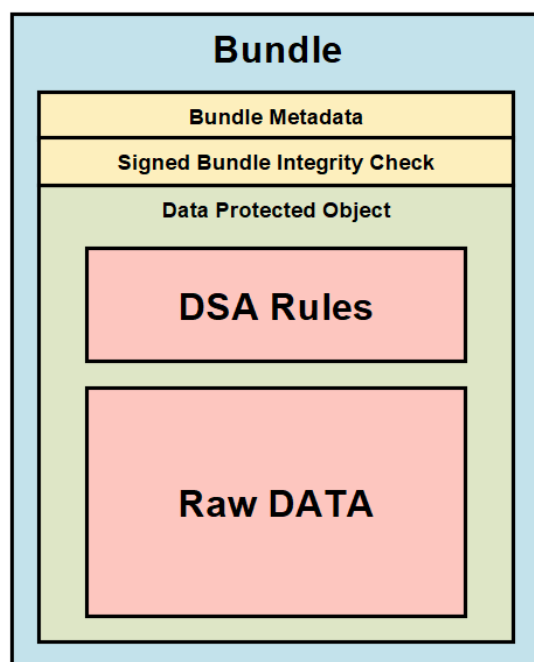


Figure 28 Data Bundle Structure

The complete description of the internal structure of the Data Bundle is given in deliverable D5.2. Once the Bundle Manager created a Data Bundle, it invokes the Bundle Store API to store the Data Bundle in the Bundle Store. The Bundle Manager should support different

encryption models depending on the distribution of the key used to encrypt data. Hence, to encrypt raw data in the Data Bundle and depending on the deployment, the Bundle Manager may use the following approaches:

- One key for each partner allows each stakeholder involved in sharing process to access the content of the data bundle by exploiting the same (private) key.
- One key for each object is used to encrypt each data bundle provided by the stakeholder using new symmetric key. In this case, the data bundle can be shared freely. However, only distinct recipients will be able to extract the key.
- One key for all objects enables the platform to encrypt multiple data bundles provided by the stakeholder with the same encryption key.

To enable encryption of the data shared by stakeholders, the Bundle Manager uses the Key & Encryption Manager of the CSI subsystem.

During the unpacking process, the Bundle Manager is used for data unpacking after getting the data bundle from the Bundle Store and performing the data decryption process when invoking the read operation on a data bundle. In particular, during the unpacking phase, the Bundle Manager retrieves the DSA paired with a specific data bundle and forwards it to the DUCS (See Section 4.2) to evaluate the access request against the retrieved DSA. Furthermore, the DSA paired with the particular data bundle may include the enforceable Data Manipulation Operation (DMO). Hence, the DUCS may invoke the DMO Toolbox (Section 4.3) to execute the specified operation on data if certain conditions are met.

4.6. *Bundle Store*

The ISI subsystem uses the Bundle Store to store data provided by the data producers in the form of the Data Bundle. At the end of the second year, this component has not undergone significant maturations or extensions compared to the version available at the end of the first year of the project. Hence, for the sake of completeness, in the following we report the description we gave in deliverable D6.1. The Bundle Store can be implemented differently depending on the implementation of ISI. Hence, for Central ISI that enables multiple Prosumers to store their data, the Bundle Store is implemented as the Hadoop Distributed File System (HDFS) [HDFS]. This implementation is suitable for big data processing, including big data analysis. However, considering the lower computational resources of the Local ISI, the Bundle Store is implemented as a protected file system. It restricts access to the user that impersonates the application server, which runs the ISI subsystem components. Hence, for this implementation, the Bundle Store is mapped to a Docker Volume, which easily permits advanced functionalities, including data storage on remote hosts or cloud providers and encryption of the content.

4.7. *Bundle Store Interface*

To enable ISI components to manage the storage of data bundles, the Bundle Store is supported with its API, which is an external interface of the Bundle Store, called Bundle Store

Interface. Furthermore, the IAI indirectly uses Bundle Store API to search and retrieve data bundles to use them in collaborative analytics via the ISI API.

The Bundle Store Interface provides the following functionalities:

- **CREATE BUNDLE:** creates a data bundle in the Bundle Store by using the file provided by the data owner with the associated DSA file, hash code and a data bundle metadata header;
- **READ BUNDLE:** allows entities to retrieve the data bundle from the Bundle Store repository according to the given ID;
- **DELETE BUNDLE:** removes the data bundle from the repository according to the corresponding ID;
- **SEARCH BUNDLE:** queries the metadata of the data bundle repository and returns a set of metadata entries corresponding to the matching data bundles according to a given JSON-based search string.

The data bundle metadata is used by the ISI subsystem to classify and search each data bundle and to enable collaborative analytics on that record according to enforceable DSA paired with the uploaded data. The metadata used in the ISI subsystem are divided into metadata related to the dataset provided by data owners and stored in the data bundle (e.g., Start Time, End Time, Event Type), and additional metadata that describe attributes specific to the E-CORRIDOR network (e.g., ID, DSA ID). While stakeholders may specify metadata for each dataset, additional metadata is inserted by Bundle Manager and Bundle Store to distinguish it from other data bundles stored by the ISI subsystem. It is worth noting that the metadata related to the provided dataset may change during the creation process of a bundle. For example, if data associated with the data bundle is changed by the enforcement of one or more DMOs specified in the corresponding DSA, then the corresponding metadata will change as well. Meanwhile, the metadata will remain static for the data bundle lifetime once it is stored in the Bundle Store.

Table 5 Data Bundle MetaData Example

Data Bundle Metadata Example
<pre>{ "id": "12345", "dsa_id": "54321", "start_time": "2021-03-22T09:00:00.0Z", "end_time": "2021-03-22T10:00:00.0Z", "event_type": "NIDS Event", "organization": "CNR" }</pre>

Table 5 provides an example of the data bundle metadata described through attributes. In particular, the ID uniquely identifies the Data Bundle, while the DSA ID attribute specifies

the ID of the corresponding DSA paired with the data bundle. Other attributes specify start and end time for the event represented by the Data Bundle, its type, and the name of the organization that uploaded such data.

4.8. Buffer Manager

The Buffer Manager is in charge of creating and managing the temporary data storage areas, known as Virtual Data Lakes (VDL), where the ISI subsystem prepares the data to be used for the execution of a given analytics. These data are prepared when the IAI API receives a request for executing an analytics, which specifies a set of Data Bundles on which the analytics should be executed (or, better, a query to finding them), and the IAI API invokes the method `PREPARE DATA` of the ISI API to ask for preparing such data. The produced VDL is reserved only for the execution of the request for which it has been produced, which is identified by a Session ID (returned when the analytics is launched, see Section 5.1) because the DSAs paired to the requested Data Bundle have been evaluated on that specific request to decide whether the related data can be included in the VDL or not. Moreover, as a further security measure, the data in the VDL are encrypted. We currently use symmetric encryption with a temporary key, but the Bundle Manager can be configured to use other cryptographic schemas. Obviously, the cryptographic schemas that has been used and the related temporary key must be communicated to the IAI which will need to decrypt the data in the VDL for executing the analytics.

At the end of the second year we implemented and deployed a version of the Buffer Manager which exploits a local file system to create the VDLs. We are also working on a further version of the Buffer Manager which integrates Apache Impala for data management.

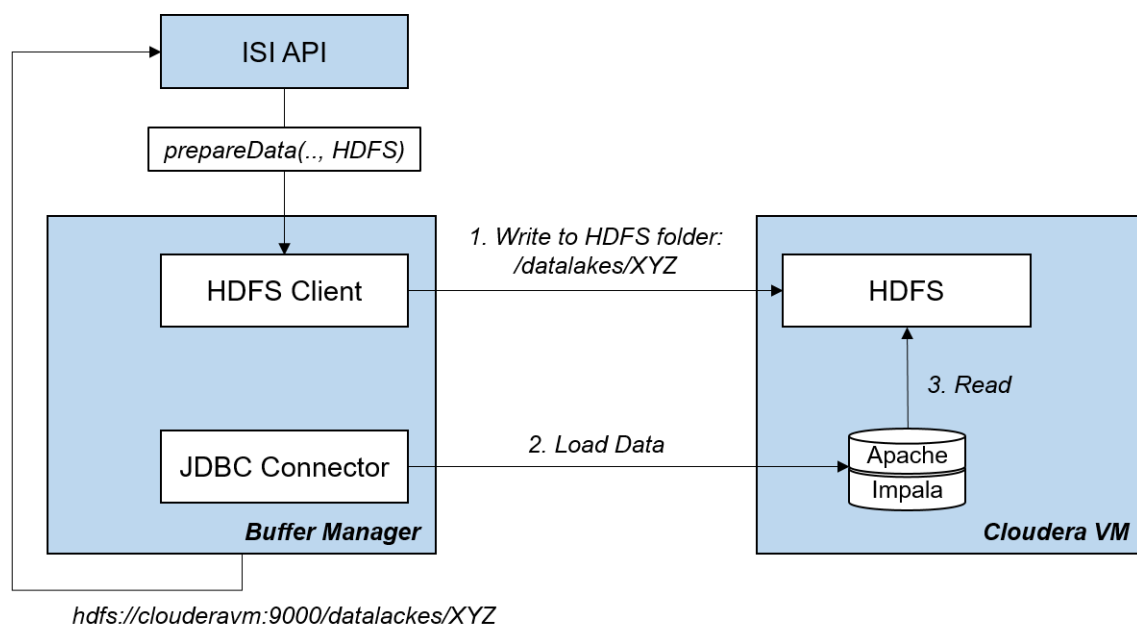


Figure 29 Write data to HDFS

Figure 29 shows the workflow between the Buffer Manager and the Apache Impala that runs on a Cloudera¹ Virtual Machine (VM) distribution. The Buffer Manager creates a VDL on the Hadoop Distributed File System (HDFS) and then loads data by the via the Java DataBase Connectivity (JDBC) driver for the Apache Impala. Finally, the Apache Impala writes the data on the ad-hoc prepared VDL.

¹ <https://www.cloudera.com/products/open-source/apache-hadoop/impala.html>

5. Information Analytics Infrastructure

The Information Analytics Infrastructure (IAI) subsystem of the E-CORRIDOR framework has the main functionality of supporting the execution of analytics on the data that have been stored in the E-CORRIDOR framework to extract new knowledge from them. A relevant feature of the E-CORRIDOR framework is that it allows to integrate a number of analytics that could have been developed separately from the E-CORRIDOR framework. In fact, the IAI allows to integrate such analytics in the E-CORRIDOR framework also after its deployment, provided that they follow the protocol we defined, by properly specifying them in the IAI configuration file. In this way, the E-CORRIDOR framework is able to satisfy the needs of the pilots use cases that could have raised subsequently its deployment.

Figure 30 shows the internal architecture and the main components of the IAI subsystem.

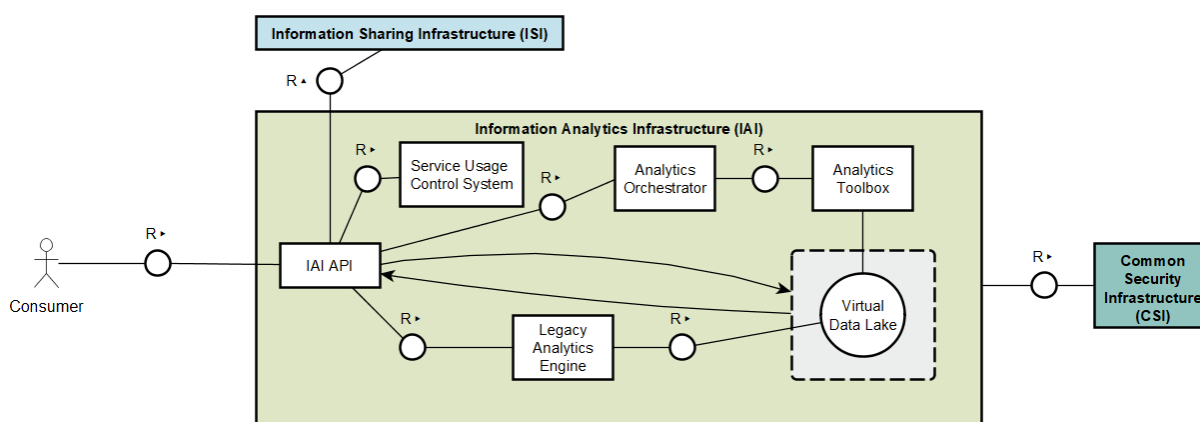


Figure 30 Information Analytics Infrastructure internal architecture

The Information Analytics Infrastructure subsystem has been heavily matured in the second year of the project. In particular, the design of the IAI API, of the Analytics Orchestrator and of the Analytics Toolbox have been finalized, and a first version of such components have been developed. The first version of the Analytics Toolbox has been integrated in the IAI subsystem.

The users of the E-CORRIDOR framework which interact with the IAI subsystem are called consumers, because they consume the data uploaded on the framework by producers (see Section 4). Such users, interacts with the IAI subsystem directly, using the APIs provided by the IAI API component, or through specific graphical interfaces, not shown in Figure 30, which, in turn, use the APIs provided by the IAI API component. The consumer specifies the analytics he/she wants to execute, and the set of Data Bundles on which such analytics must be executed. The choice of the Data Bundles is performed by imposing filters on the metadata paired with them, as explained in details in Section 5.1.

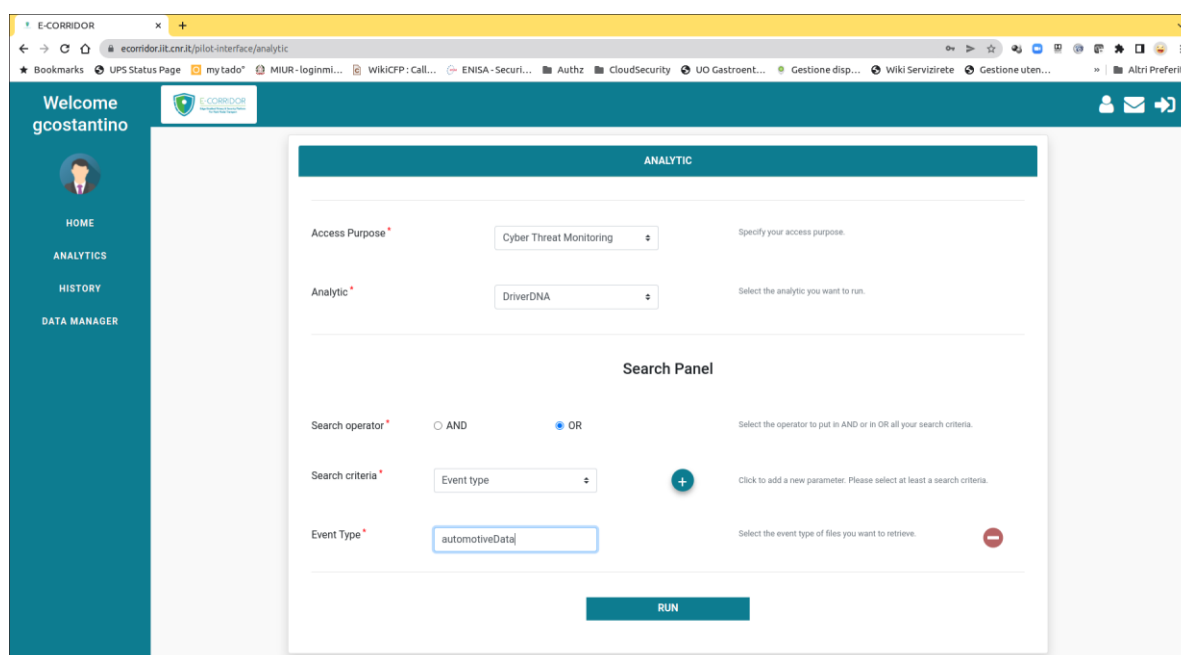


Figure 31 Graphical interface to interact with the E-CORRIDOR framework

Figure 31 shows an example of a graphical interface to interact with the E-CORRIDOR framework. In particular, the screenshot shows the page to launch an analytics. This interface is generic, and it has been designed for testing purposes. We recall that each pilot will have its own graphical interface for interacting with the E-CORRIDOR framework, which will be customized according to the pilot's need.

Before launching the execution of the analytics, the IAI API performs the authorization process, by invoking the Service Usage Control System (described in Section 5.2), which evaluates the Service-level Usage Control Policy to check that consumer is authorized to invoke such API interface in such access context, i.e., to invoke the execution of a given analytics.

If the Service Usage Control System allows the consumer to execute the requested analytics, the IAI API invokes the PREPARE DATA interface of the ISI subsystem to trigger the creation of the Virtual Data Lake, i.e., the set of Data Bundles that, according to their DSAs, can be used for the execution of the requested analytic function. To enhance the data protection level, the Virtual Data Lake is symmetrically encrypted with a temporary symmetric key, and it is destroyed after the execution of the analytic it was created for. Then, the IAI API component invokes the Analytics Orchestrator (described in Section 5.3) which coordinates the actual execution of the analytic functions on the Virtual Data Lake by properly invoking the Analytics Toolbox for the execution.

The Analytics Toolbox is aimed at supporting the integration in the IAI of analytics functions developed by third parties. The idea is that those analytics could be integrated when the E-CORRIDOR framework is deployed, or even during its lifecycle, still with a minimal effort for their integration. To allow their integration in the framework, the analytics functions must be developed and deployed following some constraint, i.e., they must be RESTful services following a given protocol and running within docker containers. The Analytics Toolbox will then maintain a registry of the analytics that have been deployed and of the URLs where they

can be invoked. Section 5.4 will describe the Analytics Services and the Analytics Toolbox in details.

Since the analytics execution could be long lasting, the protocol between the consumer and the IAI is asynchronous. Hence, when the execution of the analytics has been authorized, the IAI subsystem returns a ticket (i.e., a unique ID identifying that session) to the consumer as result of the analytics execution request. Then, the consumer will interact with the IAI again to ask whether the analytics is terminated and to get the ID of the Data Bundle embedding the results. As a matter of fact, when the results of the analytics is ready, the IAI API invokes the CREATE interface of the ISI API for creating a new Data Bundle including them. The actual access to such Data Bundle will be executed by the consumer by invoking the READ interface of the ISI API, either directly or using the pilot specific graphical interface.

Finally, the IAI, through the IAI API component, is also in charge of receiving the requests of interrupting the execution of an analytic from the ISI subsystem or from the Service Usage Control System in case of violation of, respectively, the DSA of one of the Data Bundle involved in the analytic or the Service-level Usage Control policy.

5.1. IAI API

The IAI API component is the frontend of the IAI subsystem, which is invoked by the E-CORRIDOR consumers to execute the analytics provided by the E-CORRIDOR framework. Figure 30 shows that this component directly interacts with the E-CORRIDOR consumer. Typically, as also shown by our pilot use cases, the consumer will interact with the IAI API through a customized graphical interface easing the task of selecting the set of Data Bundles on which executing the analytics and of choosing the analytics to be executed. Figure 31 shows an example of such graphical interfaces which, instead, has been developed for testing purposes.

Being the frontend of the IAI, the IAI API, oversees the authentication and the authorization of the user requesting the analytics execution. For triggering the authorization process, the IAI API invokes the Service Usage Control System (described in Section 5.2), which evaluates the Service-level Usage Control Policy to check that the invoking user is authorized to invoke such API interface in such access context, i.e., to invoke or to ask for the result of a given analytics.

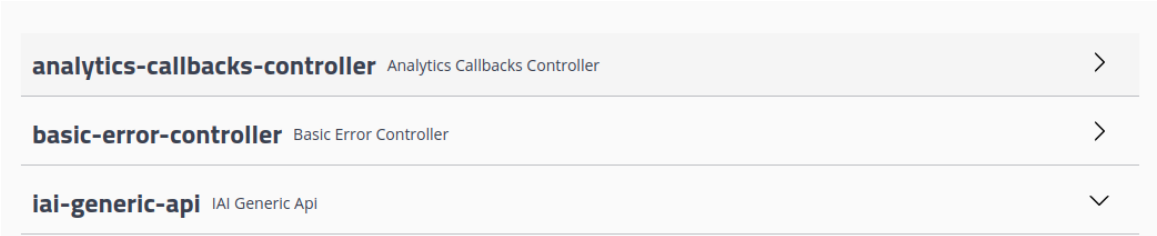


Figure 32 IAI REST API

Figure 32 shows the current version of the Open API regulating the interactions with the IAI subsystem. The API is organized in three parts: the IAI GENERIC API, providing the

interfaces for interacting with the IAI, the ANALYTICS CALLBACK CONTROLLER, providing the interface to allow the analytics service to return their results to the IAI, and the BASIC ERROR CONTROLLER.

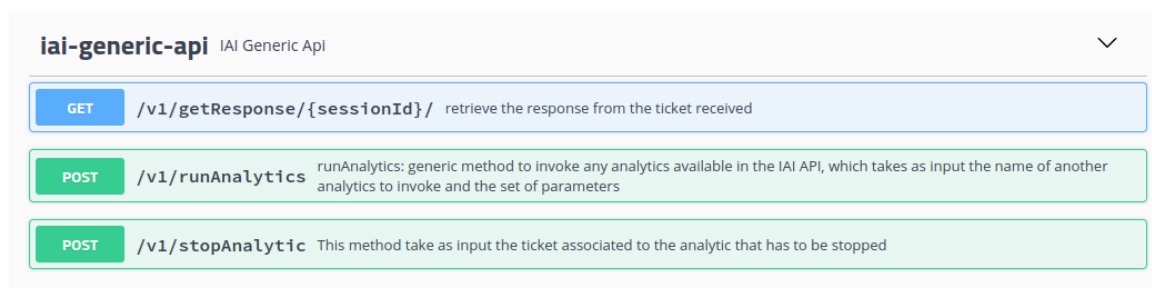


Figure 33 IAI REST API: generic API

The IAI GENERIC API, shown in Figure 33, provides a RESTful interface exposing the following methods:

- **RUN ANALYTICS:** This method allows E-CORRIDOR consumers to invoke the execution of a specific analytics on a set of data. The parameters that the consumer should submit to the IAI API when invoking this method are:
 - **PARAM:** is a string which embeds all the parameters that are passed to the runAnalytics operation

The parameters are embedded in a JSON string and they are the following:

- **SERVICE NAME:** this is the unique ID (string format) of the analytics that the consumer wants to execute. As previously explained, the IAI subsystem has been designed to allow the integration of third party analytics, i.e., analytics that have been developed independently from the E-CORRIDOR framework, provided that these analytics follow a set of constraints that are described in the following (see Section 5.4). In particular, in order to integrate these analytics in the IAI subsystem, the E-CORRIDOR framework administrator must define a unique ID for each of them, which will be used internally to refer to such analytics. For instance, “driverDNA” is the string representing the unique ID of an analytics we integrated in the current version of the E-CORRIDOR framework. The E-CORRIDOR framework administrator must list all the analytics currently provided by the framework in a YAML configuration file, called `iai_analytics.yaml`. The following is an excerpt of such configuration file concerning the Driver DNA analytics:

```
iai:
  analytics:
    .....
```

```

- id: driverdna
  name: Driver DNA
  type: http
  target: http://iai-dna:5000/
.....

```

The format of the `iai_analytics.yaml` configuration file will be explained in details in Section 5.4. The consumer can choose the analytics he/she want to execute from that list. The idea is that the graphical interface of each pilot use case, which is aware of the goals and of the functionality of the analytics, should parse this file and propose to the consumer a user-friendly graphical interface for performing the analytics invocations.

- **SERVICE PARAMS:** this is a list of parameters that must be passed to the analytics. These parameters depend on the specific analytics and could be very different from one analytics to another. Some analytics (such as the current version of the Driver DNA analytics which have been taken as example in this chapter) could even have an empty set of parameters. The IAI is agnostic with respect to these parameters, because it does not know their meaning and their usage. Here they are represented as strings, and they will be converted by the specific analytics service in the real format. Hence, the IAI simply forwards these parameters to the service implementing the analytics, which knows their formats and meaning.
- **SEARCH CRITERIA:** each analytics must be executed on a set of Data Bundles, in order to extract the new knowledge from them. Hence, the consumer invoking the analytics must specify the set of Data Bundles on which the analytics should be executed. The simplest solution would be to specify the list of Data Bundles IDs in the analytics invocation. However, to make the IAI API more flexible, we decided to allow the consumer to submit queries that will be executed by the Bundle Manager on the Bundle Store in order to find the set of Data Bundle that are relevant for the execution of the specific analytics. The **SEARCH CRITERIA** parameter is used to submit such query to the IAI subsystem which, in turn, will forward it to the ISI subsystem. These queries must involve the metadata paired with the Data Bundles. The Data Bundle metadata that are currently available are:
 - **ID:** represents the unique ID of the Data Bundle, that is defined by the ISI subsystem whe the Data Bundle is created
 - **DSA_ID:** represents the ID of the DSA which is paired with the Data Bundle
 - **START_TIME, END_TIME:** are used to represent the time period the data embedded in the Data Bundle refer to. For instance, if the data embedded in the Data Bundle is the log of a service, the **START_TIME, END_TIME** represent the beginning and the end of the interval of time the log refers to.

- **EVENT_TYPE**: represents the type of event the data embedded in the Data Bundle refers to. For instance, a possible type could be the log of a SSH service, or the video taken in a public place. Each pilot use case would define its own event types.
- **FILE_EXTENSION**: represents the format of the data embedded in the Data Bundle. This metadata does not concern the type of information the data represent, but it only specify which is the format used for representing such information. For instance, CSV, JSON, XLSX are possible formats.
- **STIXED**: this metadata specifies whether the data are in STIX format.
- **ORGANIZATION**: represents the organization the Data Bundle belongs to. Example of organizations could be CNR or HPE.

Further metadata could be added in the next version of the E-CORRIDOR framework if required by the pilot use case. A very simple example of query using the previously described metadata is the following, which specify the IDs of the Data Bundles:

ID = 1647623199140-3e353367-81bb-430a-9978-bc6d663e133e

OR

ID = 1504730198645-3e865465-8bbb-c56a-90d8-45bc6da453ea

Another example of query that can be provided as search criteria is the following:

ORGANIZATION = CNR

AND

EVENT_TYPE = SSH_LOG

where the Data Bundles we want to retrieve to be processed by the analytics are the ones produced by the CNR organization and that embed data representing logs of SSH systems

- **ADDITIONAL ATTRIBUTES**: this parameter is used to pass to the IAI further attributes concerning the analytics request that could be taken into account by the Data Usage Control System or by the Service Level Usage Control System. The purpose for which the analytics is requested, called ACCESS-PURPOSE, is an example of such additional attributes. These additional attributes are not forwarded to the service implementing the analytics.

An example of parameters used for the invocation of the RUN ANALYTICS method to request the execution of the Driver DNA analytics is the following:

```
{
  "serviceName": "driverdna",
```

```

"serviceParams": {},
"searchCriteria": {
  "combiningRule": "or",
  "criteria": [
    { "attribute": "id",
      "operator": "eq",
      "value": "1647623199140-3e353367-81bb-430a-9978-
bc6d663e133e"},
    { "attribute": "id",
      "operator": "eq",
      "value": "1504730198645-3e865465-8bbb-c56a-90d8-
45bc6da453ea"}]
  },
  "additionalAttribute": {"accessPurpose": "Cyber Threat
Monitoring"}]
}

```

Since the analytics invocation strategy is asynchronous, the RUN ANALYTICS method returns as soon as the analytics execution has been launched and returns as result one of the following values:

- 200: request successfully processed. This means that the IAI correctly elaborated the request. If the authorization phase, both at data level and a service level, state that the consumer is authorized to invoke the analytics on (a subset of) the requested Data Bundles, the RUN ANALYTICS method returns the Ticket that will be subsequently used by the consumer to check whether the results are ready. If, instead, the consumer does not have the right to exploit the requested analytics or to use all the Data Bundles he/she requested, the RUN ANALYTICS method returns a message saying that the consumer is not allowed to invoke that analytics or is not allowed to elaborate the requested data
- 500: which means that an error occurred. In this case, the Analytics service returns a JSON payload with the description of the error occurred, as follow:

```

{
  "error": "....."
}

```

- **GET RESPONSE:** This method allows E-CORRIDOR consumers to query the IAI about the status of an analytics execution request they previously submitted. The parameter that the consumer should submit to the IAI API when invoking this method is only one:

- SessionID: is the Ticket received by the consumer from the IAI when he/she submitted the analytic execution request. This ID is passed as suffix of the URL of the method

The GET RESPONSE method returns as result one of the following values:

- 200: request successfully processed. This means that the IAI correctly elaborated the request, and the response includes the following fields.
 - STATUS: this field indicates whether the current status of the analytics execution. The returned STATUS is:
 - STARTED: if the analytics execution has been started but not has not been finished yet
 - FINISHED: if the analytics execution has successfully finished and the results are ready
 - ERROR: if there were error in the execution of the analytics
 - RESULT_MESSAGE: this field is a string representing a message produced by analytics as result of the request. The IAI has no control on this string.
 - RESULT_DATA_BUNDLES: this is an array of Data Bundle IDs. These Data Bundles have been created by the IAI to embed the files that have been produced by the analytics as results of the request.

An example of response is the following:

```
{
  "status": "STARTED",
  "result_message": null,
  "result_files": []
}
```

This response indicates that the analytics execution is still in progress, hence no results are available yet. In this case, the consumer should invoke the GET RESPONSE method again after a while to check whether the analytics execution has finished. Another example of response is the following:

```
{
  "status": "FINISHED",
  "result_message": "Two results have been found",
  "result_bundles": ["15ab67c23354-3e353367-81bb-430a-8bab-45b476a4569a",
    "1646230877081-1f75b4d9-8667-4238-8e41-e81b12a0a4a1"]
}
```

This response indicates that the analytics execution is finished and two Data Bundles have been created by the IAI to embed two files returned by the analytics as results. The message produced by the analytics is represented by the string: "Two results have been found", which has no meaning for the IAI, and will be used by the software managing the specific pilot use case.

- 500: which means that an error occurred. In this case, the Analytics service returns a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

- **STOP ANALYTIC:** This method allows to stop an analytics that is currently being executed. The parameter required by this method is:
 - **SessionID:** is the Ticket received from the IAI when he/she submitted the analytic execution request.

The STOP ANALYTICS method returns as result one of the following:

- 201: request successfully processed. This means that the IAI correctly interrupted the analytics that was running.
- 401: unauthorized: This means that the requestor was not authorized to stop the analytics.
- 404: not Found: This means that the ticket provided as parameter to this method was not valid, i.e., there was no analytics running having assigned such ticket.

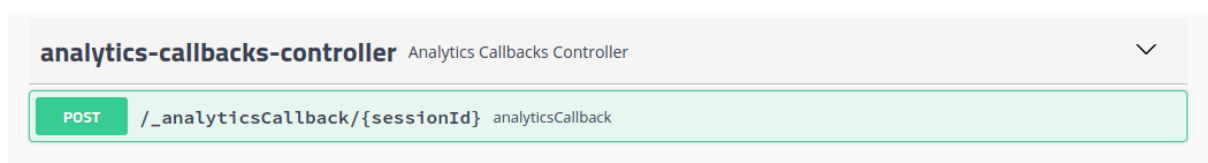


Figure 34 IAI REST API: analytics callback controller

Figure 34 shows the IAI API, focusing on the ANALYTICS CALLBACK CONTROLLER part. One method only is included in this part, which provides the interface to allow the analytics service to return their results to the IAI. More details on the usage of this method will be provided in Section 5.4.

5.2. *Service Usage Control System*

The main aim of the Service Usage Control System is to protect the IAI itself by regulating its usage by the E-CORRIDOR users according to the model defined in [CDLMM2016], which has been exploited for the Data Usage Control System as well.

The version of the Service Level Usage Control System we delivered at the end of the first year already provided almost all the required functionalities. Hence, the functionalities, the internal architecture, and the workflows of the Service Usage Control System have not undergone to relevant changes in the second year of the project. Consequently, the description reported in the following is very similar to the one presented in the previous version of this deliverable, deliverable D6.1.

This component has been integrated in the IAI because it provides a further protection of the E-CORRIDOR framework with respect to the Data Usage Control System. As a matter of fact, the difference between the Data Usage Control System that is embedded in the ISI subsystem and the Service Usage Control System is that the former protects the data that are shared through the E-CORRIDOR framework according to the policies (DSA) defined by the data producers, while the latter protects the E-CORRIDOR framework enforcing a policy defined by the infrastructure provider. In other words, if a user wants to execute an analytic on a set of Data Bundles, the Service Usage Control System will protect the E-CORRIDOR framework controlling whether the user is allowed to execute such analytic on the IAI according to the service usage control policy, while the Data Usage Control Service will protect the data Bundles controlling whether the user is allowed to use such Data Bundles according to the DSAs paired to them.

The policy enforced by the Service Usage Control System is written in UPOL. This policy is defined by the services administrators who directly upload them on the Service Usage Control System.

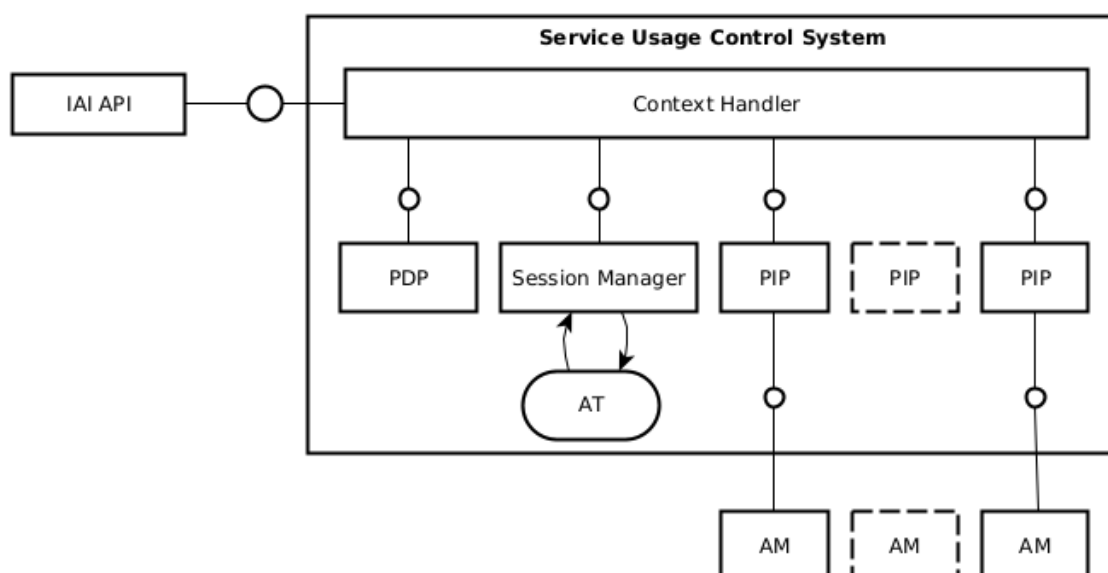


Figure 35 Service Level Usage Control System Internal Architecture

For what concerns the architecture, the Service Usage Control System is an instance of the Usage Control System described in Section 4.2.1, as shown in Figure 35. Since the users who access the IAI are obviously the same users who access the ISI, the user attributes that are used at service level are the same that are used in the DSA (a list is available in Table 4) and, consequently, the PIPs and the AMs that are used in the Service Usage Control System are inherited from the Data Usage Control Systems. Resource attributes, instead, are obviously different because they involve the IAI service instead of the data. Hence, specific AMs are exploited for retrieving such attributes and specific PIPs are developed to interact with such AMs. Examples of resource attributes are:

- The current workload of the server;
- The number of executions that are currently in progress;
- The current battery charge level (in case of mobile devices);
- The result of the integrity check made on the code installed on the server.

The Service Usage Control System is invoked by the IAI API component before executing any other operation. If the result of the policy evaluation is positive, the IAI API component retrieves the data on which executing the analytic from the ISI subsystem. Instead, in case of a negative decision, no further action is executed by the IAI subsystem, and an error message is returned to the user who invoked the analytic execution.

5.3. *Analytics Orchestrator*

One key component of the IAI subsystem is the analytics orchestrator. Its main goal is the composition of multiple analytics in a single workflow. The analytics orchestrator has been designed by taking into account the pilots' requirements and the need to run multiple analytics to effectively execute some user stories. One prominent example of these requirements concerns the exploitation of different analytics available in the IAI toolbox and executed at different touchpoints of the passenger journey for providing authentication, security, or ancillary services.

The overall idea of the analytics orchestrator revolves around managing the available analytics as building blocks to define new and more complex services. Once the workflow is specified, the composition of these services is transparently managed by the orchestrator and the data consumer perceives such a workflow just as a new simple analytics. On the other hand, thanks to the modular approach brought in by the orchestrator, developers of E-CORRIDOR analytics are relieved of any complexity inherent to monolithic design of complex services and can more easily update their components.

In the following, the main functionalities and design choices performed for the orchestrator during the second year of execution of the E-CORRIDOR project are recalled. Then, the implementation status at the time of writing this deliverable is reported as well as the plan for the final integration and testing in the pilots.

Design choices and functionalities: By examining the need for a simple workflow definition and a broader architecture support (in terms of both hardware and software), after having examined several off-the-shelf tools for orchestration (e.g., Puppet [PUPPET], Chef [CHEF],

Ansible [ANSIBLE] and Nornir [NORNIR]) our choice fell on SaltStack [SALT]. Salt is an open-source project for infrastructure management supporting remote execution and configuration management enabling hybrid cloud environments. The software is written in Python and distributed with the Apache License 2.0.

After some customization efforts, the Salt functionalities exploited by the IAI orchestrator are the following:

- Composition of analytics (deployed as software containers) in parallel and or sequential fashion
- Simple workflow definition in YAML format or Python language. The IAI orchestrator supports only a static definition of the workflows
- Support for heterogeneous architectures: Windows, Linux, Raspbian and VMware ESXi. Pilots have expressed the need to run their analytics on a different set of devices ranging from servers (either Windows or Linux) and edge-based lightweight systems (like Raspberry Pi)
- Monitoring the correct execution of the workflow and failover capabilities

The IAI analytics orchestrator foresees the presence of a manager (called Master in the Salt terminology) and a set of workers (called Minions). The Master, that can be replicated for reliability purposes, oversees the Minions. Each Minion runs a single container corresponding to the selected analytics in the IAI toolbox. Any input required by the analytics as well as any generated output is managed by the Master according to the rules expressed in the workflow.

To ensure a proper execution and to react in a timely manner the orchestrator is also in charge of performing a continuous status check. In the event of a failure the analytics are properly restarted.

Figure 36 represents the analytics orchestrator in the IAI subsystem. Any workflow, statically defined as the parallel and/or sequential composition of the analytics available in the toolbox, constitutes a new analytics for the data consumer accessing the E-CORRIDOR framework. When the workflow is invoked, the orchestrator instantiates a master and as many workers as needed to run the analytics referenced in the workflow.

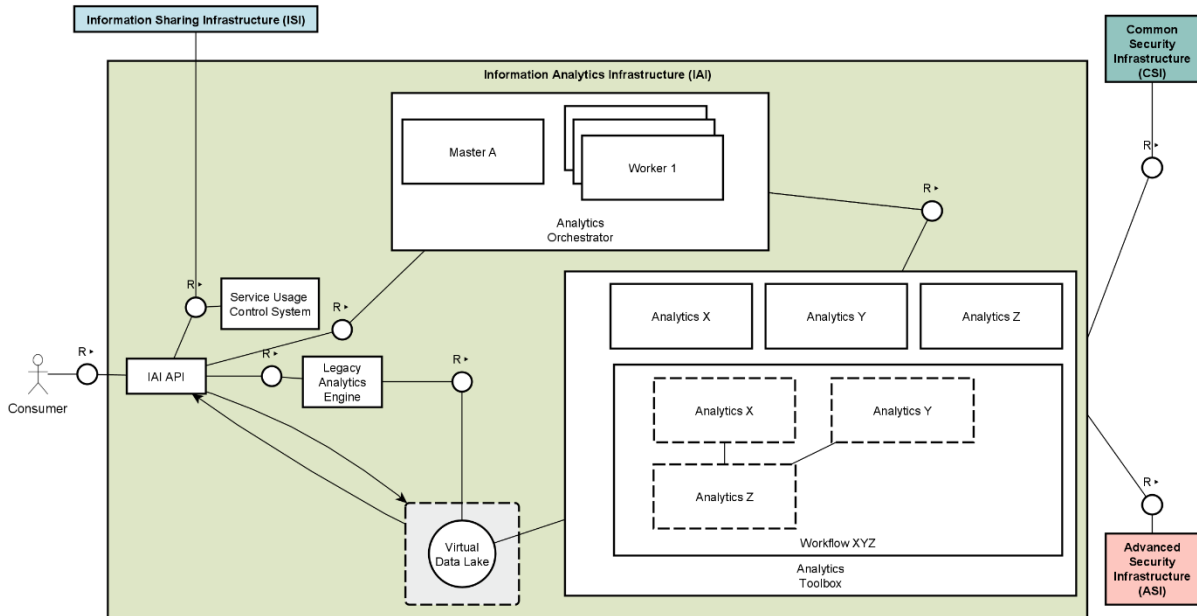


Figure 36 Analytics Orchestrator in the IAI subsystem

All the steps involved in the workflow execution are represented through the example illustrated in Figure 37. The workflow execution is invoked via REST API [step 1]. Such an execution request is sent to the proxy that appoints a Master for processing (even in a remote machine) [step 2]. One or more Minion(s) are instantiated to appropriately executed all the analytics specified in the workflow [step 3], either in parallel [step 4] or serial [step 5]. The Minion interacts with the analytics through the IAI REST API; therefore, no change is required to the analytics developers to support the inclusion of their analytics in any workflow. As soon as an analytics completes its execution [step(s) 6], the Minion returns the partial results of the workflow to its Master [step 7]. The latter processes and further instructs the Minion(s) for subsequent execution (e.g., in case of sequential execution of multiple analytics). When the workflow is completed, the Master returns the final output of the workflow [step 8].

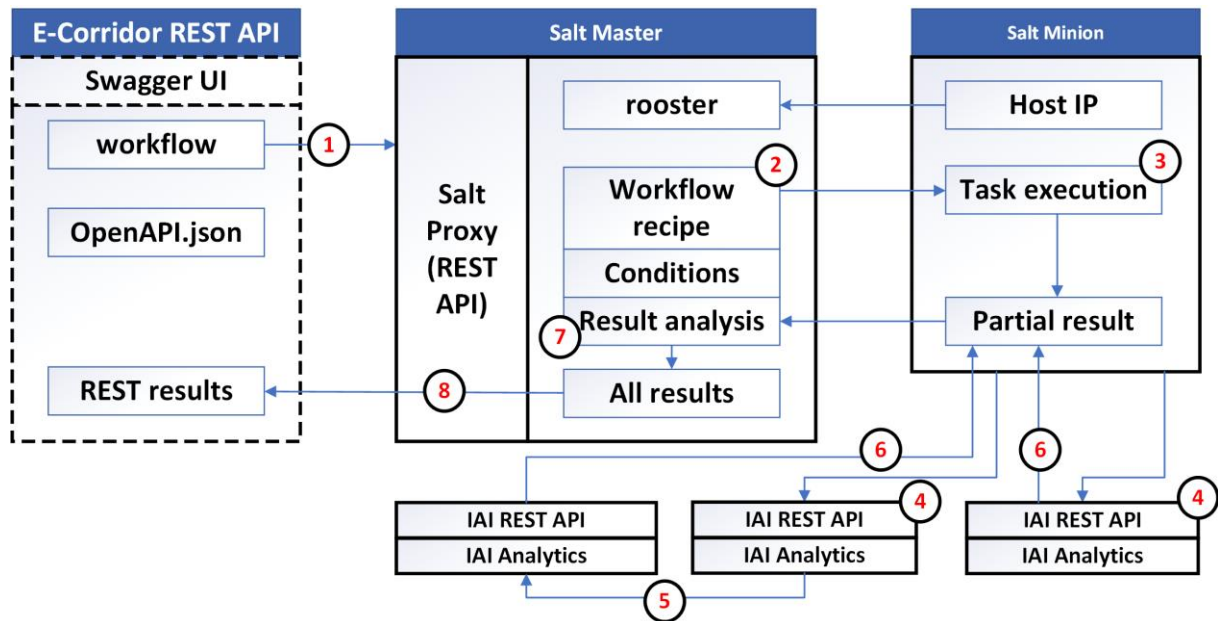


Figure 37 Example of the steps involved in the workflow execution

Implementation and test at year 2: At the time of writing this deliverable the IAI analytics orchestrator has been implemented and tested in a standalone fashion (i.e., not yet integrated in the E-CORRIDOR framework) with two analytics (whose containers were deployed in the IAI toolbox).

In particular, the development effort has targeted the following areas:

- a) REST API definition: an Open API regulating the interactions among the analytics orchestrator and the other components in the IAI subsystem. The current version of the implemented API is reported in Figure 38. It is constituted by two main parts. One dealing with the static management of the workflows available in the IAI and therefore oriented to the administrators of a node running the E-CORRIDOR framework. Instead, a second part takes care of the runtime execution of the workflows.

More in detail, the administration functions are:

- LIST: make a list of all the already created and available workflow schemas
- CREATE: specify a new workflow written as script in YAML format or python code. It is worth to remark that this function is not meant to be exposed to the E-CORRIDOR users, but it is rather made available only to administrators of the E-CORRIDOR node. This design choice aims at restricting the definition to static workflow only (i.e., to workflows whose schema is already validated and saved in the framework), as identified during the requirements elicitation
- REMOVE: remove a previously available workflow

The functions managing the runtime execution are instead:

- START: run a previously defined workflow from its schema identifier
- STOP: cancel the execution of a running workflow
- STATUS: (all or by workflow identifier): collect status information (including health check) about one or all the running workflows

Workflow schema

Workflow schema operations deals with schema creation/deletion and listing of the current schemas used by the orchestrator

^

GET	/workflow_schema/list/ List all workflow schemas available in the orchestrator	v	🔒
POST	/workflow_schema/create/ Method creates new workflow schema in orchestrator component	v	🔒
PUT	/workflow_schema/delete /{workflow_schema_id} Removes workflow schema based on existing schema ID	v	🔒

Workflow

Workflow operation to start/stop/list workflows being executed by orchestrator

Link: <https://ecorridor-at.iit.cnr.it/or...>

^

POST	/workflow/start /{workflow_schema_id} Starts new workflow based on predefined workflow schema ID	v	🔒
PUT	/workflow/stop/{workflow_id} Stop given workflow being executed by orchestrator	v	🔒
GET	/workflow/status/ Provides status of all currently executing workflows	v	🔒
GET	/workflow/status/{workflow_id} Get workflow ID execution status	v	🔒

Figure 38 Analytics Orchestrator REST API.

- b) Testing and production execution mode: the orchestrator has been designed to support two different modes of workflow execution. In the *predefined mode*, the workflow is already described in YAML or Python, stored in the IAI and only the parameters (if any) to run it are passed. This is how the workflows are meant to be executed in production. A *testing mode* is instead used to run the workflow “on the fly” during its definition to ease the initial testing, customization (e.g., starting from an already available workflow schema) and validation phases to satisfy the target use case.
- c) Deployment: the analytics orchestrator has been containerized to ease its integration and management within the E-CORRIDOR framework.

At the time of writing this deliverable the orchestrator has been tested standalone. The experiments have considered a scenario designed in the AT pilot. In such a scenario, to preserve the passengers’ privacy, face anonymization (DMO_FA – example 1 in Figure 39) is performed on the videos collected by the environmental cameras before performing their analysis (CFA – example 2 in Figure 39). From the workflow standpoint, this is an example

of sequential composition. The components considered in such a test are represented in Figure 39.

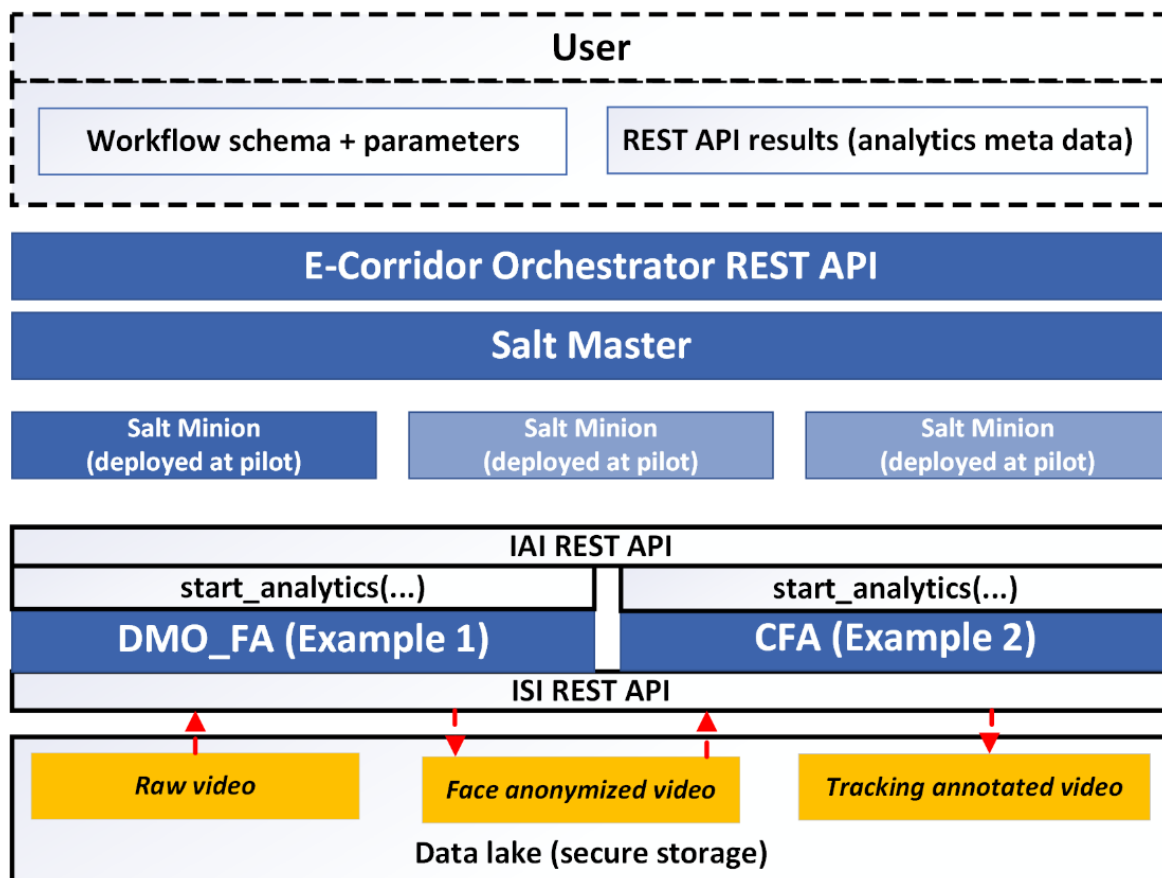


Figure 39 Components in the Analytics Orchestrator as tested in the AT pilot scenario.

A snippet of the script used to specify the sequential execution of the two analytics is reported in Figure 40. The workflow contains a section for each of the analytics and describes input and output parameters. As Master and Minions are transparently managed by the analytics orchestrator there is no need of extra configuration in the script.

```
# Start analytics #1
curl -X POST -H "Content-Type: application/json" -d '{
  "session_id": "10001",
  "iai_datalake": "None",
  "iai_dataacipher": "null",
  "iai_datakey": "null",
  "iai_files": ["/dmo/input/video1.mp4",
                "/dmo/input/vo_a1_result.avi"],
  "iai_params": {},
  "on_finish_url": "null"}' http://127.0.1.1:5000/startAnalytics --max-time 300

# Start analytics #2
curl -X POST -H "Content-Type: application/json" -d '{
  "session_id": "10001",
  "iai_datalake": "None",
  "iai_dataacipher": "null",
  "iai_datakey": "null",
  "iai_files": ["/cfa/input/vo_a1_result.avi",
                "/cfa/input/vo_a2_result.avi"],
  "iai_params": {},
  "on_finish_url": "null"}' http://127.0.1.1:5001/startAnalytics --max-time 300
```

Figure 40 Script to perform the sequential execution of two analytics

Maturation, integration, and testing plan in year 3: A few improvements are expected in the third year for the analytics orchestrator in particular regarding the definition of a simpler way for specifying new workflows. Moreover, the analytics orchestrator will be fully integrated in the E-CORRIDOR framework and its role in satisfying the user stories will be demonstrated in a few scenarios planned with the pilot(s).

5.4. Analytics Toolbox

The Analytics Toolbox is one of the key component of the E-CORRIDOR framework, since it allows to integrate in the framework a large number of analytics that could have been developed separately from the E-CORRIDOR framework, even by third parties. These analytics could be integrated in the E-CORRIDOR framework also after its deployment, in order to satisfy the needs that could have raised subsequently. This component has been designed, developed and integrated in the E-CORRIDOR framework in the second year of the project.

Analytics are data processing software components having the aim of elaborating and extracting knowledge from the data shared by the data producer in the E-CORRIDOR framework. Each analytics processes the kind of data for which it has been designed and developed. Please see deliverable D7.2 for details on the analytics designed and developed within the E-CORRIDOR project and on the kind of data they process. Further analytics could be integrated also in the third year and after the end of the project.

Any user of the E-CORRIDOR framework can behave as data producer by creating a DATA BUNDLE through the CREATE DATA BUNDLE API provided by the ISI subsystem. The data consumer, instead, is the subject who invokes the analytics provided by the IAI to obtain

new knowledge from the shared data. Data Consumers could be the same subjects who produced the data; in this case we call them *prosumer*.

The Analytics Toolbox is flexible and configurable to allow the integration of new analytics (developed either by the pilots, by the technology providers of the E-CORRIDOR project, or even by third parties) by following a *plugin* approach, thus easing the addition of new analytics even when the E-CORRIDOR framework has been already deployed, simplifying the deployment, but also protecting the source code developed by the technology providers. In particular, in order to be integrated in the E-CORRIDOR framework through the Analytics Toolbox, analytics must be implemented as https RESTful services exposing their interface by following the OpenAPI specification [OpenA2020], exposing a set of endpoints and following a protocol we defined, that is described in the following. Another requirement is that such RESTful service should be available into a docker container image, which is typically (but not necessarily) deployed on the same machine where the E-CORRIDOR framework has been deployed.

Figure 41 shows the internal architecture of the Analytics Toolbox component. On the right side of the image we can see the Analytics Services that have been developed by third parties, and that are RESTful services following the protocol we defined. The Analytics toolbox has an internal registry, the Analytics Registry, where each entry represents a registered Analytics Service and reports the related URL to invoke the service and the name to be displayed on the graphical interfaces for that analytics. This registry is populated reading the list of analytics provided in the `iai_analytics.yaml` configuration file.

Finally, the Analytics Toolbox Frontend is the subcomponent which is invoked by the IAI API to invoke an analytics service, and that reads the Analytics Registry for finding the URL to be invoked, and that operates the protocol (described in the following) to actually invoke the Analytics Service and to receive the related response and results.

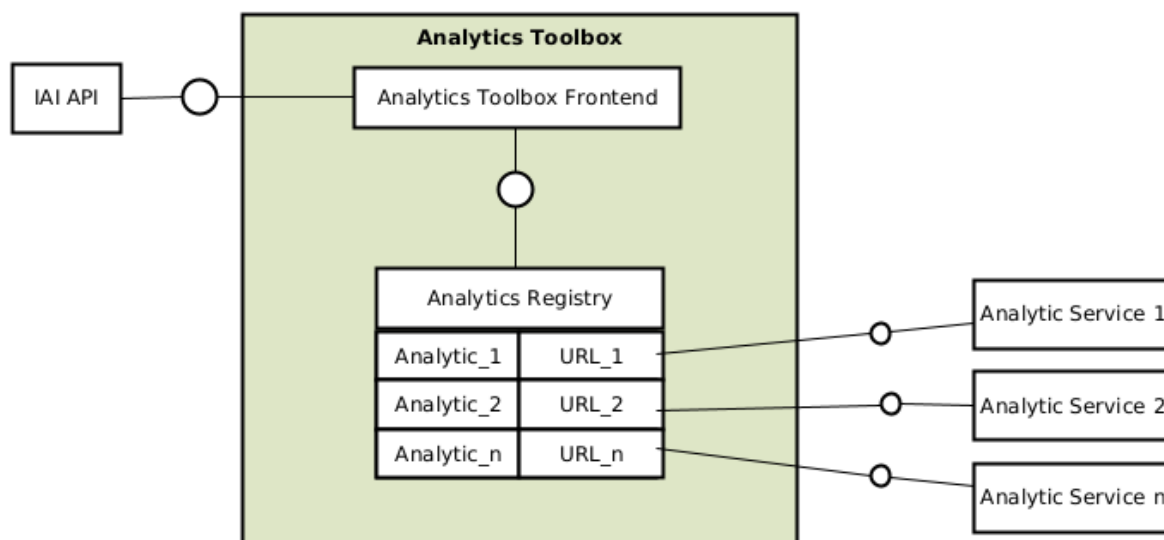


Figure 41 Analytics Toolbox internal architecture

Each Analytics Service must provide the following endpoints:

<https://0.0.0.0/startAnalytics>

<https://0.0.0.0/stopAnalytics>

The **startAnalytics** endpoint will be invoked by the Analytics Toolbox to invoke a specific analytics, choosing the proper address of the analytics service we want to execute. This endpoint is invoked through a HTTP POST method, providing the data required to run the analytics by passing the following JSON payload:

```
{
  "session_id": ".....",
  "iai_datalake": ".....",
  "iai_datacipher": ".....",
  "iai_datakey": ".....",
  "iai_params": ".....",
  "on_finish_url": ".....",
}
```

The **session_id** field reports the unique ID that has been paired with the analytics request. The **iai_datalake** field specifies the path to the temporary datalake that has been created by the ISI for running this analytics. As a matter of fact, the consumers specified a number of Data Bundle on which they want to execute the analytics, and the ISI evaluated the related DSAs, and only if the DSA evaluation result is PERMITTED the related content is copied in the datalake. Hence, this datalake contains a number of files, that have been encrypted with a temporary key for security reasons. The field **iai_datacipher** specifies the encryption scheme that has been used for the temporary encryption of the datalake, while the field **iai_datakey** represents the related decryption key. The field **iai_params** simply reports the parameters that have been passed by the consumer as input to the analytics request. The Analytics Toolbox is agnostic with respect to these parameters, whose semantic is known only by the specific analytics: the Analytics Toolbox simply forwards them. Finally, the **on_finish_url** field represents the callback URL that the analytics must use to notify the Analytics Toolbox that the analytics execution has been terminated.

The interaction protocol between the Analytics Toolbox and the Analytics is asynchronous. Hence, the Analytics simply sends back a status code as response to the startAnalytics invocation:

- 204: which means that the analytics started successfully.
- 500: which means that an error occurred. In this case, the Analytics must return a JSON payload with the description of the error occurred, as follow:

```
{
  "error": "....."
}
```

When the execution of the analytics is finished, the Analytics service must save the data representing the results, if any (e.g., pictures, videos, logs) on the datalake, and it must notify the Analytics Toolbox. To this aim, the Analytics must use the callback URL it received with the startAnalytics invocation, i.e., the endpoint analyticsEnded. In particular, the Analytics must invoke the callback URL through a HTTP POST method, sending back the results by passing the following JSON payload:

```
{
  "success": ".....",
  "value": ".....",
  "results": [".....",
              ".....",
              "....."
            ]
}
```

Where the field **success** has value TRUE if the analytics has been performed successfully, FALSE in case of errors. The **value** field is a string returned by the analytics. The Analytics Toolbox does not know the semantic of this string, it simply forwards it as textual result of the analytics. For instance, it could contain a result code, which is meaningful for an application that invoked the analytics (this is the case of the DriverDNA analytics). Another possibility is that this string contains a textual result of the analytics (e.g., “3 dangerous objects detected”). In case of error, this string could contain an error code, or a textual description of the error. The field **results** is an array containing the paths of the files that have been produced by the analytics as results. These files must be encrypted using the encryption scheme a key received in the startAnalytics request. The Analytics Toolbox simply sends back an ack message as response to the Analytics service.

To ease the integration of analytics within the E-CORRIDOR framework, we also provide an Analytics service skeleton, called IAI Agent, i.e., a RESTFUL service which implements the Analytics service in the previously described protocol. The analytic developer, which could be a third party with respect to the E-CORRIDOR framework, must simply integrate the code of its analytics in such skeleton.

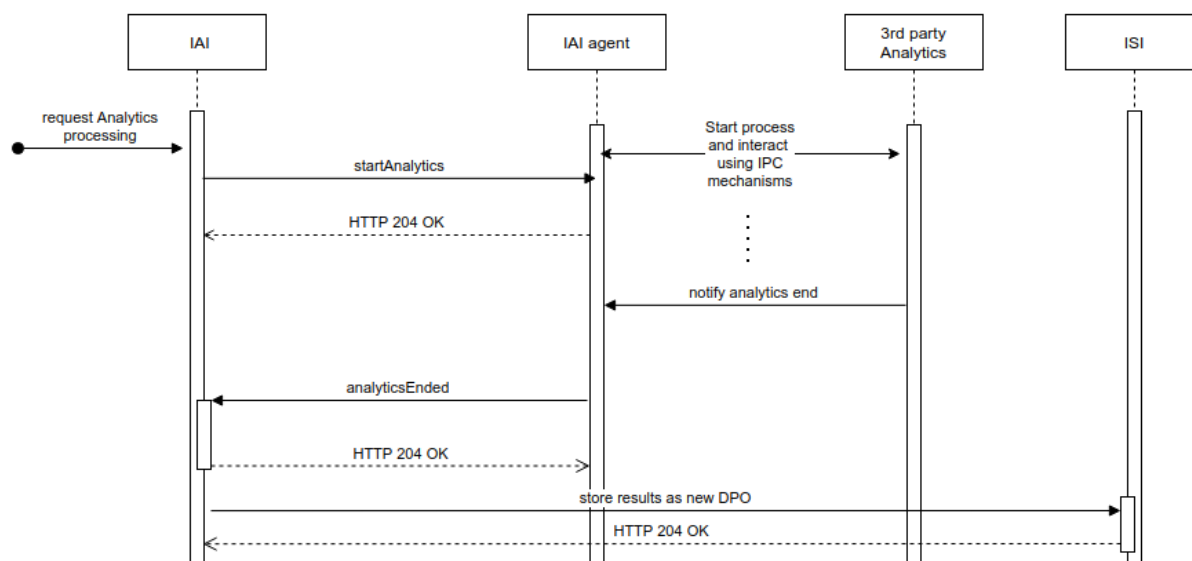


Figure 42: Sequence diagram of analytics execution

Figure 42 shows a sequence diagram representing a simple example of execution of an analytics, following the protocol previously explained. In particular, the consumer requests to execute an analytics to the IAI subsystem of the E-CORRIDOR framework (“request Analytics processing” in the Figure). For the sake of simplicity, we suppose to execute a simple analytics, hence we don’t need the orchestrator intervention here. The IAI, through the Analytics Toolbox, invokes the Analytics service (“startAnalytics” invocation in the Figure), whose frontend is implemented by the IAI agent. The IAI agent has been customized by the analytics developer to trigger the execution of the specific code of the analytics, called 3rd party Analytics in the Figure, through the Inter-Process Communication (IPC) mechanism. When the analytics code is terminated, the IAI agent invokes the “analyticsEnded” endpoint of the Analytics Toolbox exploiting the URL it previously received. The results produced by the analytics have been saved as files in the datalake, and their path in such datalake have been returned as results. Hence, the Analytics Toolbox creates one new Data Bundle for each of these files (“store results as new DPO” in the Figure), and returns the IDs of such Data Bundles as results of the analytics.

The other endpoint that must be provided by the Analytics service is the **stopAnalytics** endpoint, which, instead, will be invoked by the Analytics Toolbox to interrupt the execution of an analytics that is currently in progress. Hence, after an analytics has been started through the invocation of the startAnalytics endpoint, the Analytics service could receive a request to stop that execution. This request is sent to the endpoint through an HTTP POST method, and since the Analytics Service could run multiple instances of the analytics in parallel, the stopAnalytics request must specify the unique ID that has been paired with the analytics request that must be stopped. In this case too, the interaction between the Analytics Toolbox and the Analytics service is asynchronous, and the latter simply sends back a status code as response to the stopAnalytics invocation. The status code that are returned are the following:

- 204: which means that the stop request has been successfully received. No message is returned.
- 500: which means that an error occurred. In this case, the Analytics service returns a JSON payload with the description of the error occurred, as follow:

```
{  
  "error": "....."  
}
```

The reasons why an analytics must be stopped are mainly related to violations of the DSAs of the Data Bundles involved in the analytics, or to violations of the security policy protecting the IAI service. As a matter of fact, as previously explained, the Data Usage Control Service component running in the ISI subsystem, and the Service Usage Control Service running in the IAI subsystem continuously evaluate the DSAs and the service level policy because the factors that are taken into account in such policies are dynamic, i.e., they can change over time. Hence, if a DSA or a service level policy violation is detected while the involved analytics is still running the execution of such analytics must be interrupted. Hence, the Data Usage Control Service or the Service Usage Control Service interacts with the Analytics toolbox to ask to interrupt the analytics execution. Consequently, the Analytics Toolbox invokes the stopAnalytics endpoint, to ask the Analytics Service to (gracefully) interrupt the execution. If possible, the Analytics Service should produce an intermediate result to be returned. Otherwise, the execution is simply interrupted and no intermediate results is returned. In both cases, the Analytics Service invokes the Analytics Toolbox exploiting the callback URL received in the startAnalytics request to notify that the execution is terminated. If intermediate results are available, they are returned to the Analytics Toolbox in the same way they are returned in case of normal termination.

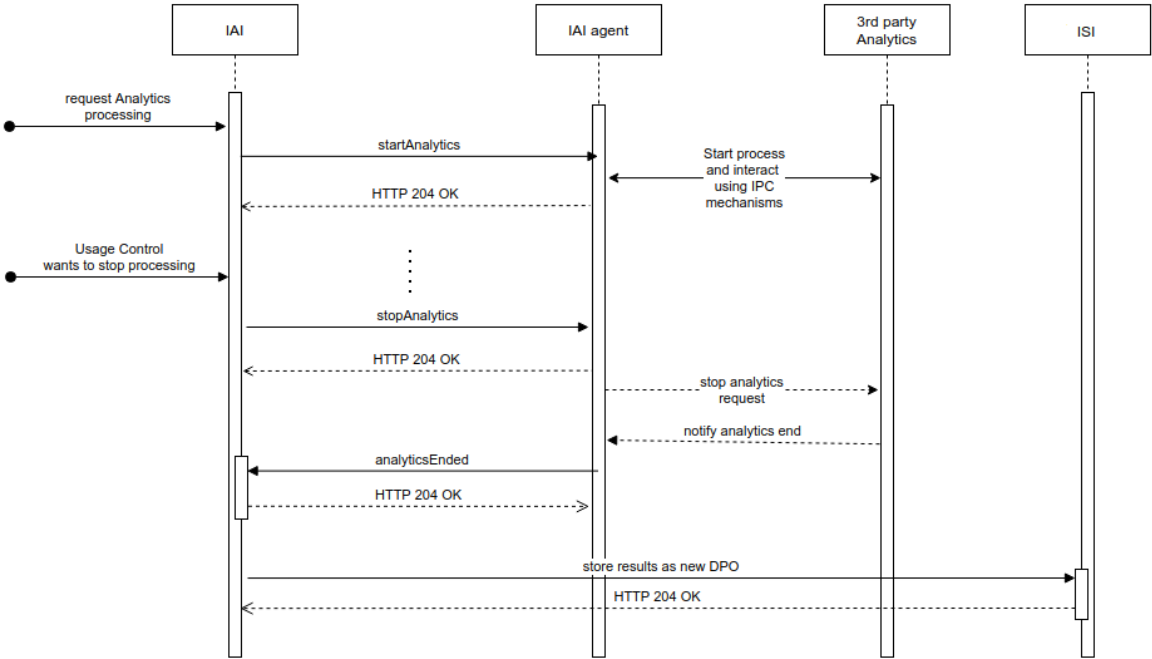


Figure 43: Sequence diagram of analytics interruption because of a policy violation

Figure 43 shows a sequence diagram representing a simple example of analytics interruption, following the protocol previously explained. The first steps of the sequence diagram concerns the invocation of the Analytics Service to start the analytics execution, and they are the same as before. Hence, the consumer requests to execute an analytics to the IAI (“request Analytics processing” in the Figure), and the IAI, through the Analytics Toolbox, invokes the startAccess endpoint of the Analytics service, whose frontend is implemented by the IAI Agent, to trigger the execution of such analytics. The IAI agent returns to the Analytics Toolbox the response code 204, meaning that the analytics execution has been started.

At this point, in this example we assume that the Data Usage Control Service running in the ISI subsystem detects a violation in one of the DSA paired with the Data Bundles used in the analytics request, and invokes the IAI to stop such execution (interaction “Usage Control wants to stop processing” in the Figure). This assumption doesn’t affect the generality of this example, because if the violation had been detected by the Service Level Usage Control Service, which runs in the IAI subsystem, we would not have had an external invocation to the IAI, but the rest of the interaction would have been the same. In order to ask the interruption of the analytic execution, the IAI, through the Analytics Toolbox, invokes the IAI agent frontend of the Analytics service through the “stopAnalytics” endpoint. The IAI agent has been customized by the analytics developer to perform the operation required to interrupt the analytics execution and, if possible, to save the intermediate results on the datalake. When the analytics execution is actually terminated, the IAI agent invokes the “analyticsEnded” endpoint of the Analytics Toolbox exploiting the URL it previously received, as it does in the normal termination case.

Like in the case of normal termination of the analytics, the paths corresponding to the files in the datalake representing intermediate results are returned in the JSON paired with the “analyticsEnded” invocation, and the Analytics Toolbox creates one new Data Bundle for each of these files (“store results as new DPO” in the Figure), and returns to the consumer who invoked the analytics the IDs of such Data Bundles as result.

The Analytics Services that have been developed following the previously described constraints can be integrated in the E-CORRIDOR framework in a very easy way. As previously stated, in order to ease the integration of analytics within the E-CORRIDOR framework, we also provide an Analytics Service skeleton, i.e., a web service already implementing an Analytics Service according to such rules. In this cases, the integration can be executed by simply adding an entry to the `analytics.yaml` configuration file of the IAI subsystem. The format of each entry is the following:

```
analytics:
  - id: analytics_ID
    name: Analytics Name
    target: https://ANALYTICS-SERVICE-ADDRESS/
```

Where the **id** field specifies the unique internal ID of the analytics, and it will be used in the DSA to refer to such analytics, the **name** field specify a human-friendly name for the analytics, while the **target** field specify the URL of the Analytics service.

5.5. *Legacy Analytics Engines*

The E-CORRIDOR framework allows the integration of existing applications which are of interest of the pilots. These programs are called Legacy Analytics Engines and they are not aware of running under the E-CORRIDOR framework. They could be deployed on the E-CORRIDOR premises, or they could be even installed on third party devices. The E-CORRIDOR framework simply invokes these applications by properly passing them the data to be used for the analytics, that have been prepared in the Virtual Data Lake. After their invocation, the E-CORRIDOR framework has no further control on the execution of legacy analytics. For instance, the E-CORRIDOR framework cannot interrupt legacy analytics in case a policy violation arises when the execution of the analytic is still in progress. For this reason, it is very important that data producers, in their DSAs, carefully define the authorization policies concerning legacy analytics, disclosing only the data field that are really necessary for the execution of the legacy analytic.

The legacy analytics are invoked by the users through the E-CORRIDOR framework, in particular through the IAI API, in order to exploit the ISI subsystem to prepare the data to be analysed according to their DSAs. Hence, the IAI API will expose a specific method for invoking each of the legacy analytics integrated in the E-CORRIDOR framework, and each of these methods will invoke the specific application implementing the legacy analytic according to its technical features (e.g., invoking a library, launching the execution of a program on a local machine, invoking a local service, invoking a remote service, etc.). Since the legacy analytics are not aware of running within the E-CORRIDOR framework, they have their specific formats for input data. Hence, the ISI subsystem and, in particular, the Buffer Manager component, should be able to create Virtual Data Lakes having the data formats required by each legacy analytics integrated in the E-CORRIDOR framework.

The development of the support for legacy analytics is still in progress at the end at the second year of the project.

6. Conclusion

This deliverable described in details the internal architecture and the functionalities of the DLI, ISI and of the IAI subsystems of the E-CORRIDOR framework, updated at the end of the second year of the project. These subsystems are at the base of the E-CORRIDOR framework, because they provide some core functionalities, i.e., the definition of DSAs and their lifecycle management, the privacy preserving data sharing among the users of the framework, and the collaborative data analytics execution.

In the second year of the project we have focused on making the E-CORRIDOR framework very general, flexible and easily configurable in order to accommodate the needs of the E-CORRIDOR pilots, and also to be easily adopted also in further scenarios involving different kind of data to be shared and different collaborative data analytics to be executed. This goal has been reached through the DMO Toolbox, the Obligation Toolbox, the Analytics Toolbox, and the Analytics Orchestrator components, which allow to easily integrate new Data Manipulation Operations, new Obligation, and also new data analytic functions (and even compositions of them) operating on several kinds of data, by simply providing them as external RESTful services (following the protocols we defined) and properly updating the E-CORRIDOR configuration files.

We are confident that this flexibility would really encourage the adoption of the E-CORRIDOR framework in further real scenarios besides the ones of the pilot use cases.

7. References

Here we provide bibliography references used in the document:

- [D6.1] Deliverable D6.1: Sharing and Analytics Infrastructure Architecture. WP6 – Information Sharing and Analytics Infrastructure. *Edge enabled Privacy and Security Platform for Multi Modal Transport (E-Corridor)*. H2020 funded project, GA: 8831135
- [D5.2] Deliverable D5.2: First Version of E-CORRIDOR Architecture. WP5 – E-CORRIDOR Platform: Requirements / Architecture / Implementation and Integration. *Edge enabled Privacy and Security Platform for Multi Modal Transport (E-Corridor)*. H2020 funded project, GA: 8831135
- [D7.2] Deliverable D7.2: Data Analytics techniques first maturation. WP7 – Data Analytics techniques. *Edge enabled Privacy and Security Platform for Multi Modal Transport (E-Corridor)*. H2020 funded project, GA: 8831135
- [D5.4] Deliverable D5.4: Final Reference Architecture. *Edge enabled Privacy and Security Platform for Multi Modal Transport (E-Corridor)*. H2020 funded project, GA: 8831135
- [KEY] Keycloak Open Source Identity and Access Management: <https://www.keycloak.org/> (last accesses May 2022)
- [DOC2021] Docker, OS-level virtualization, 2013, <https://www.docker.com/>
- [AVH2003] Grigoris Antoniou and Frank Van Harmelen. Web Ontology Language: OWL. In *Hand-book on Ontologies in Information Systems*, pages 67–92. Springer, 2003.
- [WHO22] <https://www.who.int/westernpacific/emergencies/covid-19/information/physical-distancing> (last accessed Feb 2022)
- [DLMMM2018] F. Di Cerbo, A. Lunardelli, I. Matteucci, F. Martinelli, P. Mori: A Declarative Data Protection Approach: From Human-Readable Policies to Automatic Enforcement. *WEBIST (Revised Selected Papers) 2018*: 78-98
- [XACML2013] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0, January 2013. 21, 52, 53
- [STIX] Structured Threat Information eXpression (STIX): <https://stixproject.github.io/> (last accessed May 2022)
- [PS2004] J. Park and R. Sandhu. UCON_{ABC} usage control model. *ACM Transactions on Information and System Security*, 7 (1) (2004), 128–174
- [CDLMM2016] E. Carniani, D. D'Arenzo, A. Lazouski, F. Martinelli, P. Mori. *Usage Control on Cloud Systems*. *Future Generation Computer Systems* 63(C): Elsevier Science (2016), 37-55
- [SCFY1996] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman: Role-Based Access Control Models. *Computer* 29(2): 38-47 (1996)
- [HKFV2015] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo and J. Voas, "Attribute-Based Access Control," in *Computer*, vol. 48, no. 2, pp. 85-88, Feb. 2015
- [OPENA2020] OpenAPI Specification, OpenAPI Initiative, 2020, <http://spec.openapis.org/oas/v3.0.3>

- [BAL2021] WSO2 Balana implementation: <https://github.com/wso2/balana> (fetched April 2021)
- [HDF2006] HDFS, Hadoop Distributed File System (HDFS) a distributed file system designed to run on commodity hardware, 2006, <https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>
- [PUPPET] Puppet Workflow: https://puppet.com/docs/pe/2019.8/orchestrator_workflow.html
- [CHEF] Chef Infrastructure: https://docs.chef.io/chef_overview/
- [ANSIBLE] Ansible: <https://docs.ansible.com/ansible-tower/latest/html/userguide/workflows.html>
- [NORNIR] Nornir: https://nornir.readthedocs.io/en/latest/plugins/execution_model.html
- [SALT] SaltStack: <https://docs.saltproject.io/en/latest/topics/orchestrate/index.html>
- [SWA2001] Swagger, an Interface Description Language for describing RESTful APIs expressed using JSON, 2001, <https://swagger.io/>