E-CORRIDOR
Edge Enabled Privacy & Security Platform For Multi Modal Transport

D6.1

Sharing and Analytics Infrastructure Architecture

WP6 – Information Sharing and Analytics Infrastructure

E-CORRIDOR

Edge enabled Privacy and Security Platform for Multi Modal Transport

Due date of deliverable: 31/05/2021
Actual submission date: 31/05/2021

31/05/2021
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, I. Matteucci, G. Giorgi, F. Martinelli (CNR), S. Sebastio (UTRC), M. Manea (HPE)

Approved by: C. Abdelgani (UPEC) and R. Orizio, P. Sobonski (UTRC)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	16-Mar-2021	P. Mori	CNR	Initial ToC
0.2	19-Mar-2021	P. Mori	CNR	Sections 5 and 5.1: IAI
0.3	25-Mar-2021	P. Mori, O. Oleksii	CNR	Sections 4, 4.1, 4.2, 4.5, 4.6, 4.7, 4.8
0.4	30-Apr-2021	I. Matteucci, P.Mori	CNR	Sections 3.4, 4.3, 4.4
0.5	1-Apr-2021	G. Giorgi	CNR	Contribution to sections 4.3, 4.4
0.6	2-Apr-2021	S. Sebastio	UTRC	Section 5.4
0.7	9-Apr-2021	P. Mori	CNR	Section 6: Requirements Analysis
0.8	14-Apr-2021	S. Sebastio	UTRC	Contribution to sections 4.3, 4.4
0.9	19-Apr-2021	P. Mori, M. Manea	CNR, HPE	Section 1: Introduction, Section 2: Architecture overview
0.10	23-Apr-2021	P. Mori, I. Matteucci	CNR	Revised Section 1, added Section 3.1: DSA and CNL
0.11	28-Apr-2021	O. Oleksii	CNR	Revised and extended sections 4.6, 4.7, 4.8
0.12	29-Apr-2021	F. Martinelli	CNR	Section 7
0.13	30-Apr-2021	P. Ciampoli	HPE	Section 3, 3.2, 3.3.
0.14	02-May-2021	S. Sebastio	UTRC	Section 5.3
0.15	10-May-2021	M. Ammara	CLEM	Contribution to sections 4.3, 4.4
1.0	31-May-2021	P. Mori	CNR	Addressed comments from reviewers. All sections involved.

Executive Summary

Deliverable D6.1 is the first output of Working Package 6, Information Sharing and Analytics Infrastructure. The main aim of this deliverable is to describe the first version of the internal architectures of the E-CORRIDOR infrastructures in charge of providing the privacy preserving sharing capability, the **Information Sharing Infrastructure (ISI),** and the analytics capability, i.e., **Information Analysis Infrastructure (IAI)**. These two infrastructures are at the base of the E-CORRIDOR framework, and they are deployed on the devices building up an E-CORRIDOR installation for a specific pilot, ranging from powerful servers to mobile devices, according to the deployment models defined in deliverable D5.2. Moreover, this deliverable also describes a third subsystem of the E-CORRIDOR framework, the **Data Sharing Agreements Lifecycle Infrastructure (DLI)**, because such subsystem is very related with the previous two since it allows to produce and manage the **Data Sharing Agreements (DSA)**, i.e., the privacy preserving policies that are enforced by the ISI subsystem when data are used for executing collaborative analytics by the IAI subsystem.

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 this deliverable. Moreover, this deliverable is focused on the internal architectures and functionalities of the ISI, IAI, and DLI, while the interactions among the five subsystems of the E-CORRIDOR framework to implement the E-CORRIDOR framework functionalities are described in Deliverable D5.2.

Finally, in order to validate the architectures of the three previously mentioned subsystems, this deliverable examines the requirements concerning the E-CORRIDOR framework that have been defined in deliverable D5.1. In particular, this deliverable analyses whether and how the internal architectures and functionalities defined for the DLI, for the ISI, and for the IAI subsystems contribute to satisfy each of the framework requirements defined in deliverable D5.1.

Table of contents

List of Figures

1. Introduction

This deliverable is the first output of Working Package 6, Information Sharing and Analytics Infrastructure, whose main objective stated in the E-CORRIDOR's Description of Work (DoW) "is the development and integration of tools and technologies for both the **Information Sharing Infrastructure (ISI)** and **Information Analysis Infrastructure (IAI)**", which are the subsystems of the E-CORRIDOR framework meant to provide two main functionalities:

- The Information Sharing Infrastructure **regulates the data sharing** among different parties enforcing specific access and usage control policies paired with such data called **Data Sharing Agreements (DSA)**.
- The Information Analytics Infrastructure allows E-CORRIDOR users to **execute data analytics functions** exploiting the data shared through the ISI, obeying to the sharing and analytics constraints expressed in the DSAs paired with such data.

The E-CORRIDOR's DoW states that:

"*E-CORRIDOR's mission is to define a framework for multi-modal transport systems, which provides secure advanced services for passengers and transport operators. The framework includes collaborative privacy-aware edge-enabled information sharing, analysis and protection as a service*".

From the previous sentence, it is clear that the controlled data sharing and the analytics capabilities provided by the ISI and by the IAI subsystems play a central role in the E-CORRIDOR framework.

Since the controlled data sharing provided by the ISI is implemented by enforcing specific access and usage control policies that are paired with the data itself, the DSAs, the E-CORRIDOR architecture includes a third subsystem, called **DSA Lifecycle Infrastructure (DLI)**, which oversees managing the life cycle of such DSAs, from their creation to their deletion.

Hence, this deliverable describes in detail the internal architectures of the DLI, ISI, and IAI subsystems, which have been produced as outputs of Tasks, respectively, T6.1, T6.2, and T6.3, as well as the interactions among the internal components of such subsystems. Instead, the overall interactions among all the subsystems of the E-CORRIDOR architecture (which are 5 in total) to implement the workflows required to address the pilots use cases, are described in deliverable D5.2.

These three subsystems 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 focussing on sharing and analysing a very specific set of data, i.e., Cyber Threat Information (CTI), E-CORRIDOR generalizes these concepts and ideas to a fully-fledged framework for sharing and analysing any kind of data (also enabling analytics at the edge). As a matter of fact, the E-CORRIDOR pilots, which are focussed on multimodal transportation, take into account a very wide set of distinct data types (described in Section 3.1.1 of deliverable D5.1), e.g., Car CanBus data, GPS data, Boarding Passes, Passports, Images and Videos, Accelerometer data, Gyroscope data, Bluetooth and WiFi RSSI data, RFID data, Lidar data, Passenger data, Network logs and Event logs, airplane tracking, and many others. Consequently, the E-CORRIDOR framework must be able to manage all the data types required by the pilots. Moreover, a large number of different analytics are required by the E-CORRIDOR pilots as well (e.g., analytics for the identification of drivers and passengers, for baggage tracking, or for privacy preserving and preference based itinerary planning), thus requiring a

maturated version of the IAI where new analytics can be easily integrated without disrupting the original architecture.

For the definition of the architectures of the DLI, ISI and IAI subsystems of the E-CORRIDOR framework, we started from the requirements identified for these components in Section 3 of deliverable D5.1, called "Framework Requirements", which summarize the needs of E-CORRIDOR, taking into account its pilots. In sections 3, 4, and 5 of this deliverable, at first we describe the architectures we defined for the DLI, the ISI, and the IAI subsystems, respectively. Then, for each of the Framework Requirements expressed in D5.1, section 6 describes how the proposed architectures contribute to satisfy it.

1.1. Deliverable Structure

The rest of the 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, explaining the interfaces provided by the subsystem to create new data objects as well as to read and to use them to perform analytics. In particular, a detailed description of the security support allowing to regulate the access to, and the usage of such data is given.
- Chapter 5 describes the IAI subsystem, showing the service level security support and the capability of executing analytics which are composition of other analytics. The description of the specific analytics that will be supported by this subsystem is given in deliverable D7.1.
- Chapter 6 analyses the Framework Requirements defined in Section 3 of deliverable D5.1 to understand whether they are met by the architecture of the DLI, ISI and IAI subsystems that have been defined in this deliverable.
- Chapter 7 shows which of the project objectives defined in the E-CORRIDOR DoW are achieved through the activities reported in this deliverable.

1.2. Definitions and Abbreviations

Term	Meaning
AM	Attribute Manager
API	Application Programming Interface
CANbus	Controller Area Network
CH	Context Handler
CNL	Controlled Natural Language

CRUD	Create, Read, Update, Delete
CTI	Cyber Threat Information
DC	Decision Combiner
DSA	Data Sharing Agreement
DLI	DSA Lifecycle Infrastructure
DUCS	Data Usage Control System
eIDAS	electronic Identification, Authentication and trust Services
FHE	Full Homomorphic Encryption
GPS	Global Positioning System
HDFS	Hadoop Distributed File System
IAI	Information Analytics Infrastructure
IDS	Intrusion Detection System
IMU	Inertial Measurement Unit
ISI	Information Sharing Infrastructure
JAR	Java ARchive
JSON	JavaScript Object Notation
MRH	Multi Request Resource Handler
OBD	On-Board Diagnostics
OWL	Ontology Web Language
PDP	Policy Decision Point
PIP	Policy Information Point
RFID	Radio Frequency IDentification
RSSI	Received Signal Strength Indicator
SMS	Short Messages Service
SQL	Structured Query Language
SUCS	Service Usage Control System
TRL	Technology Readiness Level
UCON	Usage Control
UPOL	Usage Control Policy
XACML	eXtensible Access Control Markup Language

2. E-CORRIDOR Framework Architecture Overview

The overall architecture of the E-CORRIDOR framework has been defined in D5.2, and it is shown in Figure 1. The overall framework consists of 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). The first three subsystems, DLI, ISI, and IAI are described in detail in this deliverable, and in the following of this section we give an overview of the main interactions involving them, aimed at easing the understanding of their internal functionalities which will be described in the rest of this deliverable. Hence, not all the interactions among the E-CORRIDOR subsystems are listed here: the detailed description of all the interactions among all the five subsystems of the E-CORRIDOR architecture to implement the workflows required to address the pilots use cases is given in deliverable D5.2.

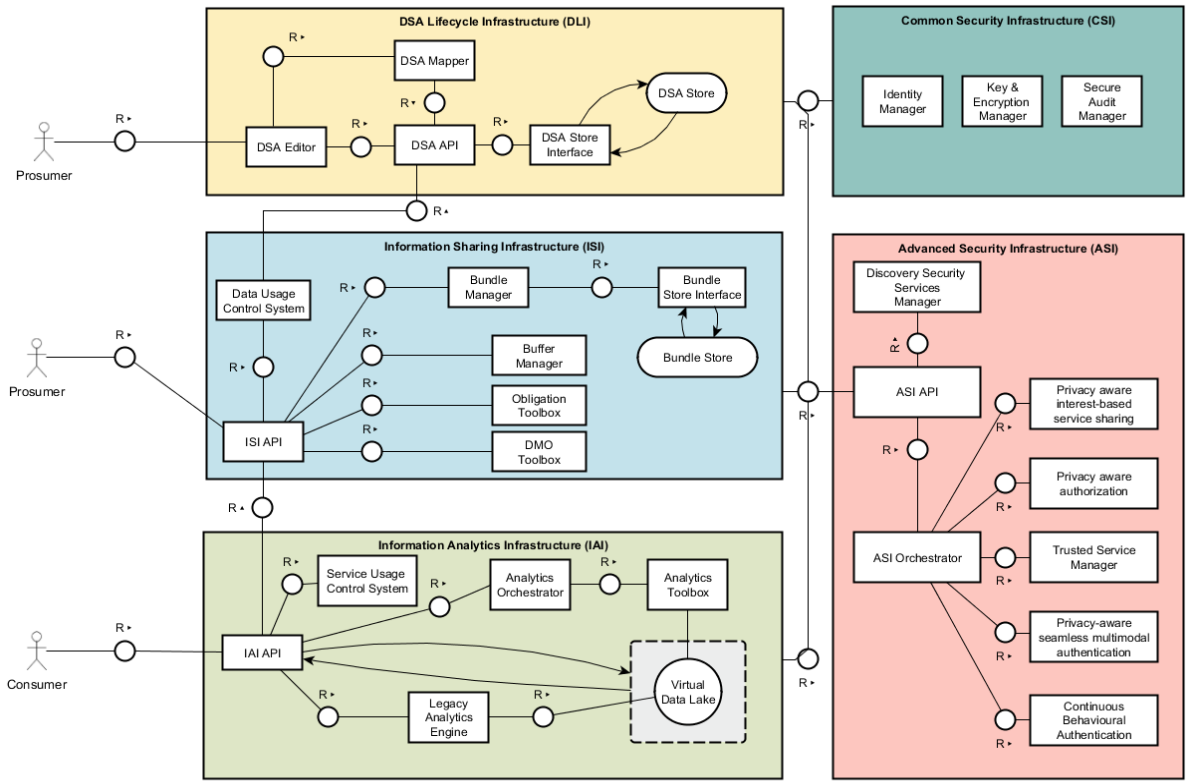

Figure 1 E-CORRIDOR framework overall architecture.

The **DSA Lifecycle Infrastructure**, described in Section 3 of this document, is the part of the E-CORRIDOR framework which allows E-CORRIDOR data producers (called prosumer, i.e., producers and consumers) to define DSAs. Hence, the users of the E-CORRIDOR framework interact with the DSA Lifecycle Infrastructure subsystem to create new DSAs through a graphical user-friendly interface, to visualize existing DSA, to update existing DSAs, or to invalidate or even delete them. This subsystem invokes the CSI subsystem to perform user authentication.

The **Information Sharing Infrastructure,** described in Section 4 of this document, is the part of the E-CORRIDOR framework which implements the confidential and privacy

preserving data sharing by providing a flexible secure storage system along with the support for the enforcement of the DSAs paired with the data. This subsystem invokes:

- the CSI subsystem to perform user authentication and to get the encryption keys for encrypting and decrypting the Data Protected Objects embedded in the Data Bundles.
- the DLI subsystem when a new data bundle is created, in order to choose and get the DSA to be embedded into this Data Bundle, and every time that a request for accessing and using a Data Bundle is received, in order to check whether the associated DSA is still valid, i.e., it has not been invalidated by the creator, and that it is not out-of-date, i.e., its version is equal to the latest one.
- the IAI to request to interrupt the execution of a running analytic when a violation of a DSA of one of the data exploited for such analytic occurs while the analytic is still in progress (i.e., results have not been released yet).

The **Information Analytics Infrastructure,** described in Section 5 of this document, is devoted to the execution of the data analytics that are required by the pilots on the data that have been stored in the E-CORRIDOR framework. Hence, the users (data consumers) of the E-CORRIDOR framework interact with the IAI to request the execution of a given analytic function on a given set of data. This subsystem invokes:

- the CSI subsystem to perform user authentication.
- the ISI subsystem to search for Data Bundles (performing queries exploiting the metadata paired with the Data Bundles), and to obtain the requested set of data on which executing the requested analytic function. Moreover, the IAI subsystem interacts with the ISI one also to store the results of the execution of a given analytics as a new Data Bundle.

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

3. Data Sharing Agreement Lifecycle Infrastructure

This section describes the DSA Lifecycle Infrastructure (DLI), which is the subsystem of the E-CORRIDOR framework that allow creation, management, and implementation of DSAs, with an introduction to DSA concepts and ontology.

The main internal components of the DLI subsystem are:

- DSA API: front-end to call DSA functions of the platform;
- DSA Editor: tool with a web Graphical User Interface (GUI) for the management of DSAs;
- DSA Mapper: translator from CNL (see Section 3.1.1) statements to programmatic instructions;
- DSA Store: it is the database where the DSAs are persisted;
- DSA Store Interface: it is the programmatic interface to the DSA Store.

Figure 2 shows the internal architecture of the DLI subsystem.

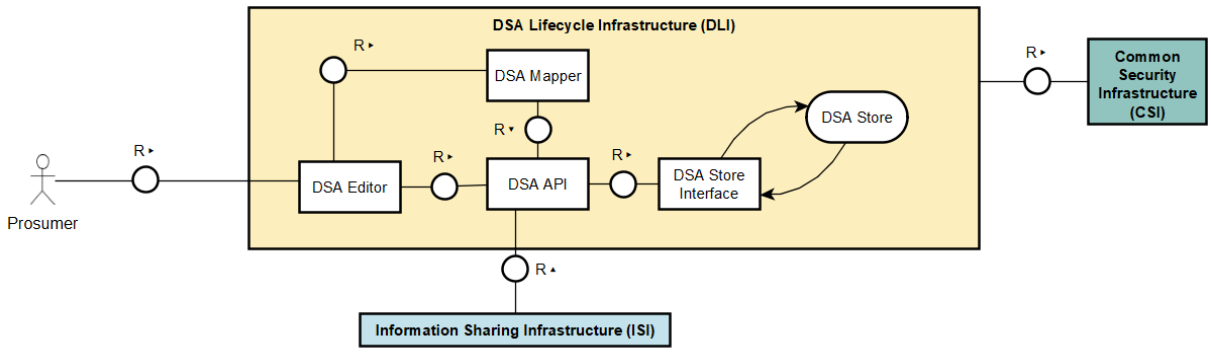

Figure 2 DSA Lifecycle Infrastructure internal architecture

The main idea is that the E-CORRIDOR user (prosumer) exploits the DSA Editor tool, through its graphical interface, to define the DSAs he/she would like to pair with the data he/she will produce. These DSAs are traduced in their enforceable version by the DSA Mapper, and are stored on the DSA Store. In this way, when the prosumer will create a new Data Bundle exploiting the ISI subsystem (see Section 4), he/she will select one the DSAs stored in the DSA Store he/she has previously created to be paired with the new Data Bundle.

In the following of this section, we first give a description of DSAs and of the language we use for expressing them, and then we describe each of the internal components of the DLI subsystem.

3.1. Data Sharing Agreements

A Data Sharing Agreement (DSA) is a digital contract which defines a set of constraints to regulate the sharing of data among organizations. The DSA is at the base of the E-CORRIDOR data protection support, since it specifies which actions can be performed on each piece of data, which subjects can execute these actions, and which other conditions should be satisfied in order to authorize the execution of such actions. In particular, a DSA consists of the following information:

a. **Title**, which is a string, chosen by the policy maker, used to identify the DSA.
b. **Status**, which specifies the current status of the DSA, i.e., whether the DSA is ready to be used or not for several reasons. The diagram of the DSA statuses, and the operations which cause a state transition are shown in Section 3.2.1.

- **Purpose**, which specifies why the data protected by this DSA are shared among the parties.
- **Application Domain**, which specifies for which of the distinct application domains supported by the E-CORRIDOR framework this DSA has been defined. At this stage, a different application domain has been defined for each E-CORRIDOR pilot.
- **Data Classification**, which specifies the nature of the data that will be protected by this DSA. A set of data classes has been defined, and further classes can be added in the future if required by Pilots. The data classes currently available are:
  a. Highly Confidential data
  b. Confidential data
  c. Public data
- **Additional Information**, which is a text where the policy maker can write additional information that could help to understand the DSA and/or the scenarios where the DSA should be adopted.
- **Validity**, which is the time interval within which the DSA is valid. After the expiration date, the access to the data protected by the DSA will not be allowed anymore.
- **Parties**, which is the list of organizations who will share their data with this DSA.
- **Vocabulary**, which specifies the terminology for the DSA, and it is defined by an ontology, written in OWL (Web Ontology Language) [AVH2003]
- **Policies**, which includes a number of rules expressing the constraints to be enforced on the data sharing. Each rule consists of three components:
  a. A condition, which is a logical formula expressing constraints over the values of the attributes paired to users, data, and environment. For instance, a condition could require that the user role attribute value is equal to “Administrator”. If this logical formula is satisfied, the effect of the rule is enforced, and the action included in the rule is allowed/denied (in case of, respectively, authorization/prohibition rule, see in the following). If the condition is not satisfied, this rule does not contribute to the access decision, and the decision depends on the outcome of the other rules of the policy.
  b. An action: each rule refers to one specific action only, and it allows/denies (in case of, respectively, authorization/prohibition rule, see in the following) such action. If there are no rules for a given actions, that action will be always denied.
  c. A flag which specifies whether the rules must be applied to “other data”. This is an innovative feature with respect to the existing approaches, and it has been defined because the E-CORRIDOR analytics are executed on sets of data instead of on a single piece of data. Hence, data producers, in order to decide whether their data can be used to execute a given analytic, could also express constraints on the other data

that are involved in such execution. The rules expressing such constraints are called rules on other data.

The rules of the DSA policy are expressed using the Controlled Natural Language (CNL) language, which is described in Section 3.1.1.

The DSA includes three kind of rules, which have different effects:

a. **Prohibition**: if (at least) one prohibition rule is satisfied, the action execution right is denied, although there are other rules which would allow the action. As a matter of fact, we apply the DENY OVERRIDE approach, where the rules which deny the access right have the precedence on the rules allowing the access right.
b. **Authorization**: if (at least) one authorization rule is satisfied and there are no prohibition rules which are satisfied, then the action is permitted, according to the DENY OVERRIDE approach.
c. **Obligation**: this kind of rules does not affect the action execution right. If this rule is satisfied, i.e., the condition expressed by the rule is evaluated to TRUE, the obligation or the Data Manipulation Operation specified in the rule is executed by the E-CORRIDOR framework.

If none of the authorization rules listed in the policy is satisfied, then the access right is not granted. As a matter of fact, we apply the DEFAULT DENY approach, where the access right is denied if it is not explicitly authorized. At least one authorization rule must be present in a DSA, otherwise the DSA will always forbid the execution of any action.

3.1.1 Controlled Natural Language

The Controlled Natural Language, CNL for short, has been developed to express Authorizations, Prohibitions and Obligations rules of a DSA in a semi natural way [MPS2010]. Rules (and set of rules, i.e., policies) are expressed in terms of subject, object (or resource), action, and environment. Similarly, the eXtensible Access Control Markup Language (XACML), the well-known, de facto, standard for access control [XACML2013], relies on similar assumptions.
The features of the four elements, i.e., subjects, objects, actions, and environment, are expressed through attributes in XACML that are listed in the vocabulary, as we said above.

To specify data sharing rules, we introduce the notion of fragment denoted as *f*, *f1*, . . ., and ranged over the set *F*. The fragment is a tuple consisting of three elements, $f = \langle s,a,o \rangle$, where *s* is the subject, *a* is the action, *o* is the object, expressing that "*the subject s performs the action a on the object o*". The terms representing the action element *a* could be instantiated from the ontology.
Work in [MPS2010] also associates a formal semantics to CNL syntax, based on a modal transition system MTS [GLT1988], making the language amenable for automated processing and analysis, see, e.g., [MPSW2011].
Usually, fragments are evaluated within a specific context. In CNL, a basic context is a predicate c that characterizes the elements of the policies, like environmental condition. Simple contexts are, e.g., temporal clauses: "within a period of ten days", or location clauses: "inside the organization". In order to describe complex agreements, contexts need to be composable.

Hence, starting from the basic context *c*, we use the boolean connectors *and*, *or*, *not* for describing a composite context *C* (ranged over the set *C*) which is defined inductively as follows:

$$C := c \mid C \text{ and } C \mid C \text{ or } C \mid \text{not } c$$

More complex expressions are generated by combining fragments. We refer to such expressions as Composite Fragments (*CF*), and we denote them as *F* (ranged over the set *CF*). We distinguish two disjoint sets of composite fragments: authorization/prohibition fragments, de- noted by F_A and ranged over the set AUTH, and obligation fragments, denoted by F_O and ranged over the set OBL.

Authorization/Prohibition Fragment
The syntax of a composite authorization/prohibition fragment is inductively defined as follows:

$$F_A := \text{nil} \mid \text{can (cannot) } f \mid F_A;F_A \mid \text{if } C \text{ then } F_A \mid \text{after } f \text{ then } F_A \mid (F_A)$$

The intuition for the composite authorization/prohibition fragment is the following:

- **nil** can do nothing.
- **can (cannot) f** is the atomic authorization (prohibition) fragment. Its informal meaning is the subject *s* can (cannot) perform the action *a* on the object *o*. can f expresses that f is allowed, but not required. Dually, cannot f expresses that f is not allowed, hence it is required that f does not happen.
- **F_A; F_A** is a list of composite authorization/prohibition fragments. The list constitutes the authorization/prohibition section of the considered DSA. Whenever one term of the list performs a f -transition, then that term evolves to the correspondent derivative.
- **if C then F_A** expresses the logical implication between a composite context C and a composite authorization/prohibition fragment: if C holds, then F_A is applied.
- **after f then F_A** represents the temporal sequence of fragments. Informally, after f has happened, then the composite fragment F_A is applied.

Obligation Fragment
The syntax of a composite obligation fragment is inductively defined as follows:

$$F_O := \text{nil} \mid \text{must } f \mid F_O;F_O \mid \text{if } C \text{ then } F_O \mid \text{after } f \text{ then } F_O \mid (F_O)$$

The intuition for the composite obligation fragment is the following:

- **nil** expresses no obligation.
- **must f** is the atomic obligation fragment. Its meaning is the subject s must perform action *a* on the object *o*. Thus, the f -transition is required.
- **F_O;F_O** represents a list of composite obligation fragments. The list constitutes the obligation section of the considered DSA. Whenever one term of the list performs a f -transition, then that term evolves to the correspondent derivative.
- **if C then F_O** expresses the logical implication between a context C and a composite obligation fragment. It means that if C holds, then F_O is required.
- **after f then F_O** represents the temporal sequence of fragments. It means that after that f is performed, then F_O is required.

An example of a simple authorization fragment written in CNL is here presented.

***If** a Subject hasRole DataAnalyst AND a Data hasType Email **then** that Subject **can** invokeSpamEmailAnalysis that Data*

The example shows what analytics can be performed by users having a specific role, specifically how a *DataAnalyst* could perform the analytic *SpamEmailAnalysis* on the data sharing the current DSA.

The readability of the policy can also be improved by restructuring the fragment:

***can** f* where *f = ⟨that Subject, invokeSpamEmailAnalysis, that Data⟩,*

as follows:

*that Subject **can** invokeSpamEmailAnalysis that Data.*

3.2. DSA Editor

The DSA Editor is the Web interactive component used for creating and managing of Data Sharing Agreements (DSAs). The tool provides an interactive approach that takes advantage of the ontology technology to guide the user in the definition of the DSA policies.

When the user is creating a policy, the application suggests (through a pop-up window) terms and actions on these terms, which are compliant with a predefined ontology (called *vocabulary*), defining the semantics of the rules. Steps to write a policy are interactively guided during the editing process. Only the relevant terms and actions are shown to the user helping her/him in defining sound rules.

The DSA Editor allows the user to define two kinds of DSAs: *i)* an "abstract" and generic version of the DSA, called DSA Template, where only a reusable set of fields and rules are written and *ii)* the effective DSA (a customised instance of the template), i.e., the one that contains all the rules to be enforced and that will be attached to the data. DSA Template acts as a starting point for creating DSAs that are specific for each business domain (e.g., airport sector) without starting from scratch. An "instance" of the DSA Template must be defined in order to obtain a DSA, as shown in Figure 3 Relation between DSA Template and DSAsFigure 3.

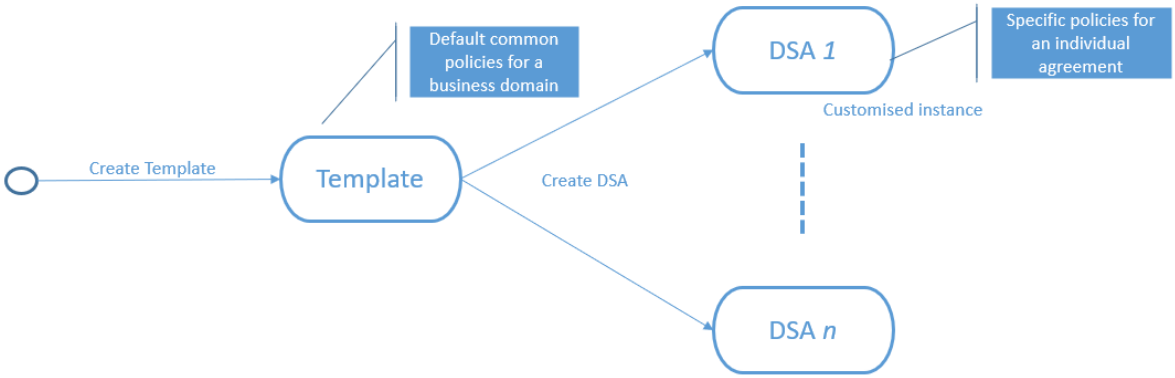

Figure 3 Relation between DSA Template and DSAs

The DSA is formalised in an XML (Extensible Mark-up Language) file format. The structure has the following main sections:

- The metadata: a unique identifier, a title, the purpose of the agreement, the temporal validity of the DSA;
- The policies which express rules about authorizations, obligations and prohibitions; they are encoded in CNL (see Section 3.1.1), based on predefined dictionaries that are defined for each distinct business domain;
- Additional data for describing (in a free-text format) the content of the DSA or to add notes.

Figure 5 shows a screenshot of the DSA Editor, while Figure 5 shows an enlarged view of the DSA authorization taken from Figure 4.

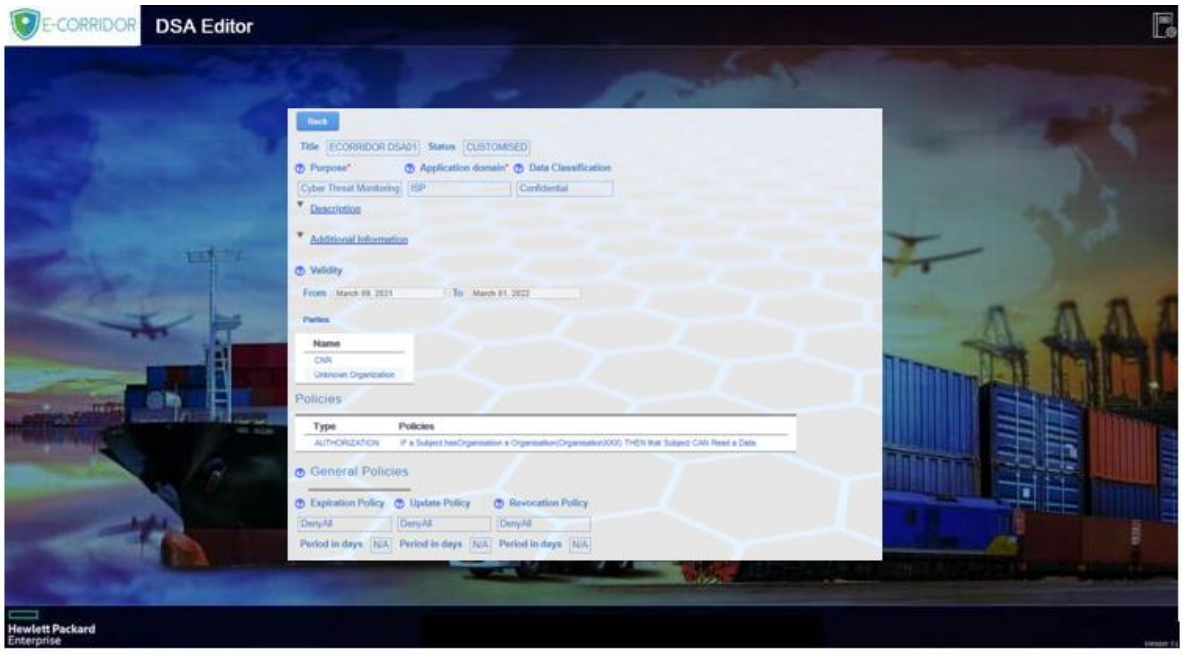

Figure 4 Screenshot of the DSA Editor

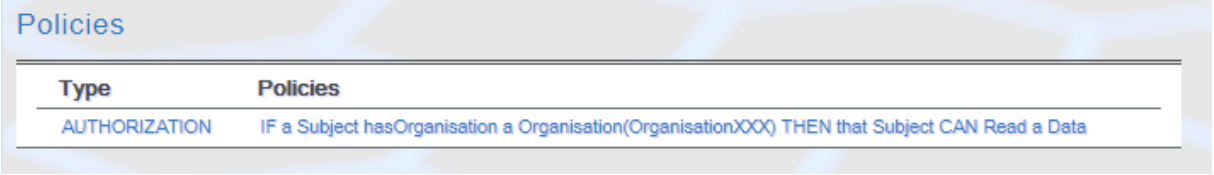

Figure 5 DSA authorization example from DSA Editor

where "OrganisationXXX" is a value specified by policy author. In this case a Subject (i.e., the user attempting a data operation), is authorised to perform the read operation on the data stored in the E-CORRIDOR platform only if she/he belongs to OrganisationXXX.

Section 3.2.2 will provide more details concerning the sentence format, where the Subject is a subclass of Term and hasOrganisation is a property with domain Subject and range a free user entry.

The DSA is then expressed as an XML file like the extract shown in Figure 6.

Figure 6 DSA XML fragment (example)

where policy is an authorization (AUTHORIZATION_1) with a set of variables: X_2 is the Subject, X_3 is the Organisation name specified in the DSA and X_4 is the data.

The vocabulary used by the DSA Editor is an ontology written in the Web Ontology Language (OWL)[1] that describes the usage context in which the DSA will be used by specifying the actions and terms and how they are related to each other.

Policies definition is made by a high-level language very close to the natural one, based on the formal language CNL (described in Section 3.1.1), that allows writing human-comprehensible sentences about the policies we want to express, yet having them "formal enough" to be processed by a machine.

The DSA Editor allows the user to define *authorisations*, *prohibitions*, and *obligations* policies: specific types of rules to express statements respectively about what an entity *can/cannot do*, *must do*, and *must not do*, as described in detail in Section 3.1.

The DSA Editor persist the DSAs (templates and instances) into the DSA Storage (see Section 3.5).

The DSA Editor supports internationalization (I18N) of its user interface for different European languages, including English (the default), Spanish, Italian and French, and can be extended to further idioms.

3.2.1 DSA Workflow and Status

At the high-level, the DSA Editor defines a two-steps editing workflow process. First a DSA called Template is created (step 1) with a predefined vocabulary described in OWL. We can consider a DSA Template as a list of predefined rules and a set of DSA Templates as a library of available rules to choose from in order to address different usage contexts or scenarios.

[1] https://www.w3.org/OWL/

Starting from a DSA Template, a DSA instance is created (step 2). The DSA instance is what we simply call DSA and inherits all the rules defined in the DSA Template; further rules can be added to the DSA instance to complete the DSA. Rules inherited from the DSA Template cannot be changed: the rationale behind that is segregation of responsibilities. DSA Templates should contain rules that must always be there (e.g., legal rules), maybe authored by a person with a legal background or a subject matter expert.

As the DSA moves in the editing workflow process, it may assume different states as shown in Figure 7.

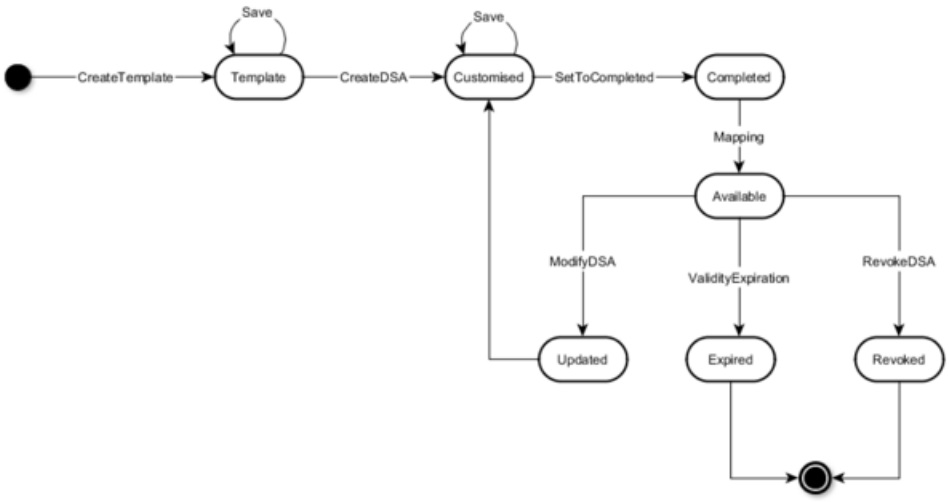

Figure 7 DSA State Diagram

At the beginning of the editing process, a DSA Template is created and set to the "Template" state. Starting from the DSA Template, further rules can be added to create a new DSA (the instance): this moves the DSA into the "Customised" state. During both editing phases, users can periodically save their work, before moving to other states.

Once the editing is completed, the user can move the DSA (instance) to the "Completed" state: it means that the DSA has been finalised and can be translated to its enforceable representation. When the DSA is translated (mapped), it is marked as "Available", ready to be selected at data ingestion time of the ISI API (see Create operation of ISI API in Section 4.1). The DSA, at this stage, is serialised in the XML file format. Only the DSAs in "Available" state can be used by at runtime by the Data Usage Control System (see Section 4.2) subsystem.

During its lifecycle, a DSA might move to the "Revoked" status, which indicates that for some reason it is no longer valid (e.g., the agreement ceased because it was break by a party). The DSA can also shift to "Expired" when it naturally expires when its validity period is exceeded. DSA in either "Revoked" or "Expired" cannot be used anymore; data that have been associated with such DSA can be are treated with respect to specific rules (see Table 1).

A DSA that is currently "Available" can be set to "Updated", if it requires to have some modifications of its policies. This might happen even if it has been already associated with data items. Also in this case, data items with such DSA have a treatment defined in a specific rule (see Table 1).

Table 1 DSA special rules for "Expired", "Updated", and "Revoked" states

Status	Possible Behaviour to already existing data with such DSA
Revoked	Deny all access to data, Delete data in a specified period
Expired	Deny all access to data, Delete data in a specified period
Updated	Deny all access to data, Delete data in a specified period

DSA Editor ontology used as vocabulary builds a basic structure for the policy definition, which is made by two classes defining the main entities of the policies domain:

- *Term*: is a subject or object of the policy
- *Action*: is a verb representing an action (e.g., create, read, or an analytic) performed by a subject on an object

Additional terms and actions must be defined as subclasses of the Term and Action classes in any vocabulary that will be created.

In the DSA Editor, the Term (and its subclasses) and the Action (and its subclasses), follows the general syntax:

a Term CAN/CANNOT/MUST Action a Term

For example:

a Passenger CAN/CANNOT/MUST Read a Data

where Passenger and Data are subclasses of Term and Read is subclass of Action.

Besides Terms and Actions, additional *properties* can be defined in the vocabulary to support conditional clauses (if-clauses) in the policies, according to the following syntax:

IF a Term hasProperty a Term THEN a Term CAN/CANNOT/MUST Action a Term

For example:

IF a Device hasPhysicalPosition posX THEN a System CAN Read a Data

Where *Device* and *System* are subclasses of Term and *hasPhysicalPosition* is a property with Domain device and Range posX.

3.2.2 Policy Definition

The DSA Editor provides the feature for Policy Definition. It is a specific section where users can define rules about authorizations, obligations and prohibitions encoded in CNL and based on the predefined dictionaries that are use case specific.

There are specific types of rules to express statements respectively about what an entity can/cannot do, must do, and must not do. Policies supports conditional clause (if) related to entity properties (e.g. for a Subject entity: Role, User Id, Organisation or Group membership).

Authorisation Policies define actions permitted to a Subject and are like:

A Subject CAN Action a Data

IF a Subject <has some properties> then that Subject CAN Action a Data

Action could be simple basic actions like Create, Read, Move, Delete or Analytics algorithms.

Obligation Policies set functions that MUST be applied when an action is invoked. Obligation can be Data Manipulation Operations (DMO), which operates on the data to hide some privacy information (e.g. anonymization of geolocation, email address, IP address, etc.) or simply notification about some event related to the Data.

These policies are like:

A System MUST Function a Data

AFTER a Subject CAN Action THEN a System MUST Function that Data

Prohibition policies define an action that a Subject cannot do:

A Subject CANNOT Action a Data

IF a Subject <has some properties> THEN that Subject CANNOT Action a Data

The DSA Editor allows also to define specific policies for the "result data" produced by the execution of an analytics service (we call the "result data" also derived object). It has the same structure and usage of shared data: Authorisation, Obligation, Prohibition.

3.2.3 Login and Roles

The DSA Editor component is a web tool reachable from an Internet browser: once connected to the DSA Editor, it requires user authentication.

In order to support the DSA lifecycle (see Figure 7 DSA State Diagram), the access and the use of the DSA Editor follows a role-based access control model (RBAC) with the following roles:

- A "legal expert", in charge of creating DSA Templates suitable for particular use cases. Typically, the "legal expert" is a person with a legal background of the context to model or is a subject matter expert;
- A "policy expert", in charge of defining the DSA instance starting from a predefined DSA Template. Typically, the "policy expert" is the person that finalize the DSA with the specific business rules.

The actions include CRUD (Create, Read, Update, Delete) operations as well as Complete (to move the DSA instance to the "Completed" state), Map (to map the DSA to its enforceable representation by invoking the DSA Mapper), and Revoke (to set the DSA into the "Revoked" state).

3.3. DSA Mapper

The DSA Mapper has the purpose of translating the DSA policies, defined through the DSA Editor and 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 API

The DSA API is the external interface of the DSA Lifecycle Infrastructure subsystem. It provides functions for managing the DSAs. The DSA API is used by the ISI for interacting with the DLI when it needs to retrieve DSAs or by the DSA Editor when it needs to create or update DSAs.

We have the following interfaces:

Table 2 DSA API

Group	API	Input	Output
DSA CRUD	Create DSA	DSA metadata, policies and properties	DSA identifier
	Retrieve DSA	DSA identifier	DSA
	Update DSA	DSA identifier, updated DSA data	Operation result (success or failure)
	Delete DSA	DSA identifier	Operation result (success or failure)
DSA UPOL CRUD	Add/Update enforceable policies in a DSA	DSA identifier, UPOL	Operation result (success or failure)
	Fetch enforceable policies from the DSA	DSA identifier	UPOL

	Delete enforceable policies from a DSA	DSA identifier	Operation result (success or failure)
DSA Status	Retrieve the status of a DSA	DSA identifier	DSA status
	Check DSA Validity	DSA identifier	DSA validity
	Revoke a DSA	DSA identifier	Operation result (success or failure)

3.5. DSA Storage

The DSA Store is a database where the DSAs are stored and consumed by the other components of the DLI. We foresee for this database, that will contain documents (DSA are XML files), the use of a NoSQL database like MongoDB. MongoDB uses the concept of collections, which is a group of documents, much like the same a relational database has tables and records (tuples).

The DSA Store will also be exposed, through the DSA API (see 3.4), to the ISI subsystem to retrieve the appropriate DSA for the data to be shared and protected at E-CORRIDOR Framework operation time.

For storing a DSA into the DSA store, a pair of two corresponding JSON documents are created in the DSA Store: one is the DSA itself and another is the DSA metadata, which are linked by the DSA-Id. The DSA metadata contains a set of DSA fields extracted from the DSA XML files and it is used for the sake of Search DSA. DSA Editor is able to exploit the searching on a set of DSA fields.

3.6. DSA Store Interface

The DSA Store Interface is the external interface of the DSA Store subsystem, and it provides functions for managing the storage of DSAs. The DSA Store is used by the ISI to retrieve DSAs or by the DSA Editor when it needs to create or update DSAs.

The DSA Store Interface supports the following methods:

Table 3 DSA Store Interface

Group	API	Input	Output
DSA CRUD	Create DSA	DSA metadata, policies, and properties	DSA identifier
	Read DSA	DSA identifier	DSA
	Update DSA	DSA identifier, updated DSA data	Operation result (success or failure)
	Delete DSA	DSA identifier	Operation result (success or failure)

DSA Search	Search DSA	Search criteria	A set of DSA Identifiers corresponding to the matching DSAs

Given a search criteria flag (simple or extended), the “DSA Search” operation can return:

- A set of DSA IDs (flag=simple)
- A set of entries corresponding to the matching DSAs (flag=extended)

In the latter, the set of entries contains a subset of the DSA metadata fields (e.g. DSA title, DSA purpose, etc.) for each returned DSA.

4. Information Sharing Infrastructure

The ISI subsystem implements the data collection and usage enforcement functionalities of the E-CORRIDOR framework. Hence, this is the component of the E-CORRIDOR framework which is devoted to implement the confidential and privacy preserving data sharing by providing a flexible secure storage system along with the support for the enforcement of the DSAs paired with the data. To protect data when they are at rest, they are embedded in a cryptographic container, called Data Bundle, along with their DSAs before being stored on the storage system. The Data Bundle also includes some metadata, which are stored in clear (not encrypted) and are used in the search operation. The DSA enforcement is continuous over time, according to the Usage Control model, i.e., the DSA is evaluated both at access request time, to decide whether a given action (i.e., data read and delete, or the execution of an analytics) can be executed, and also "continuously" while the actions is in progress, to decide whether the execution of such action 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 subject 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 a Data Bundle 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 that Data Bundle 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 policy is violated, and the analytic execution must be interrupted (or suspended, waiting that the user enters the airport again).

Figure 8 shows the internal architecture of the ISI subsystem.

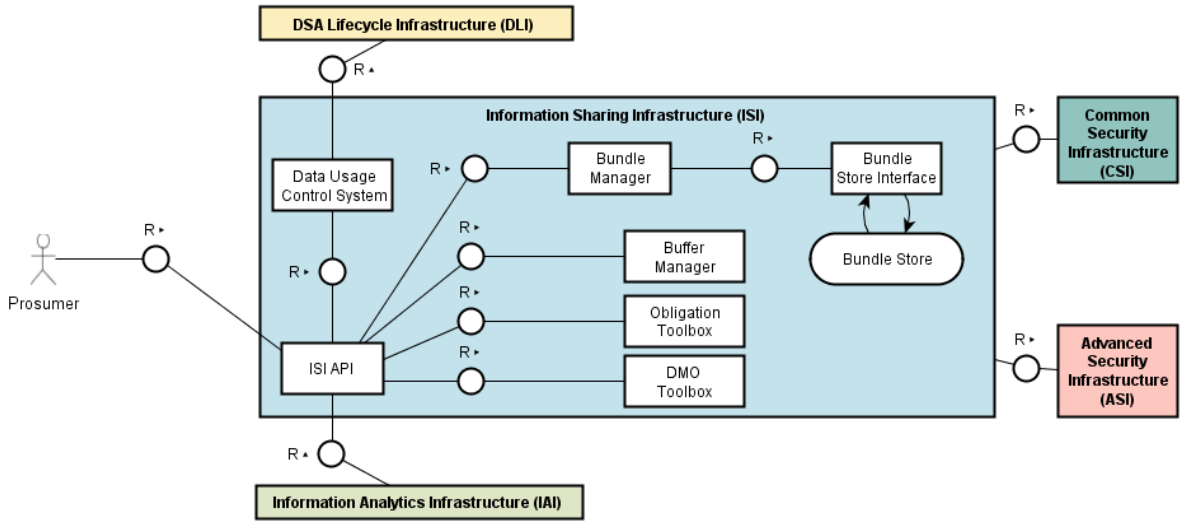

Figure 8 Information Sharing Infrastructure internal architecture.

The ISI subsystem is invoked both by the users of the E-CORRIDOR framework and by the IAI subsystem. A user invokes the ISI subsystem either using a pilot specific graphical interface or exploiting the APIs of the Information Analytics Infrastructure subsystem, which are provided by the ISI API component (described in Section 4.1). The user interacts with the Information Sharing Infrastructure subsystem to perform the CREATE, READ, and DELETE

operations on the data. Instead, for invoking the execution of analytics on the data, the user interacts with the IAI subsystem which, in turn, interacts with the ISI subsystem on behalf of the user to collets the data required for the analytics execution. The IAI invokes the ISI exploiting the ISI API. The ISI interacts with the DLI to retrieve the DSA required to create new Data Bundles.

The main functionality of the ISI is the enforcement of the DSAs embedded in the Data Bundles to decide whether the requested operations can be performed on the related data.

The requests are received by the ISI API, which is the frontend of the ISI subsystem. The ISI API invokes the Buffer Manager (described in Section 4.7) to create a Virtual Data Lake where the data whose usage have been authorized by their DSAs will be stored after that the DMOs have been performed on them. Since the data are in clear once copied in the Virtual Data Lake, the whole Virtual Data Lake is encrypted using the symmetric scheme with a temporary key.

The Bundle Manager (described in Section 4.5) invokes the Data Bundle Storage to retrieve all the Data Bundles listed in the request, extracts the related DSAs, and invokes the Data Usage Control System to check them before making the related data available on the Virtual Data Lake.

The Data Usage Control System selects the Data whose DSAs authorize their usage, and also determines the Obligations and the DMOs that must be performed on them before making them available for the execution or the requested operation.

The ISI API, after having invoked the execution of the DMOs on the data, returns the link to the Virtual Data Lake and the temporary encryption key as result of the operation over a secured channel.

The Sequence Diagrams showing the complete workflow of the Data Bundle creation and of the data analytics execution, which involves all subsystems of the E-CORRIDOR framework, are shown in Section 9 of deliverable D5.2.

4.1. ISI API

The ISI API component is the frontend of the ISI subsystem and it exposes to the E-CORRIDOR users the interface to invoke the operations create, search, read, and delete Data Bundles, and to transfer Data Bundles from an ISI instance to another. Figure 8 shows that E-CORRIDOR users interacts with the ISI API for invoking the execution of such operations. Instead, for the execution of the analytics, E-CORRIDOR users should invoke the IAI API (see Section 5.1), which, in turn, invokes a specific method of the ISI API, called prepareDataLake, to prepare a Virtual Data Lake including the data on which the analytic will be executed.

Being the frontend of the ISI subsystem, the ISI API oversees the authentication of the user requesting data access, exploiting the CSI subsystem.

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

- CREATE DATA BUNDLE: creates a new Data Bundle starting from a data file, a DSA, and a set of metadata. This method returns the ID paired with the Data Bundle.
- SEARCH DATA BUNDLE: returns the IDs of the Data Bundles whose metadata satisfy the query passed as input.
- READ DATA BUNDLE: allows to read the raw data embedded in a Data Bundle. The user submits the ID paired with the Data Bundle and the method returns the data in the

Data Bundle. The DSA is enforced to check whether the user executing the read method has the right to read the requested Data Bundle. Moreover, the DSA could also require the execution of a DMO on the data before releasing them to the user.

- PREPARE DATA LAKE: creates a Virtual Data Lake that will be used to perform an analytic. This method takes as input a number of Data Bundle IDs, and it enforces the DSA to check that all these Data Bundle can be used to perform the requested analytic. The DMO stated in the DSAs are executed on the data before releasing the Virtual Data Lake. The Virtual Data Lake is encrypted with the symmetric scheme using a temporary key. The Virtual Data Lake reference and the temporary symmetric encryption key are returned as results of the method.
- DELETE DATA BUNDLE: this method deletes a Data Bundle whose ID is passed as input. The delete operation is executed only if it is allowed by the DSA paired with the Data.
- MOVE DATA BUNDLE: this method move a Data Bundle from an ISI instance to another.

4.2. Data Usage Control System

The **Data Usage Control System (DUCS)** is aimed at regulating the usage of the data that are shared in the E-CORRIDOR framework for the execution of collaborative analytics, following the Usage Control (UCON) model [PS2004]. In particular, the DUCS is an extension of the **Usage Control System** presented in [CDLMM2016]. 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 9. Moreover, the UCON model also introduces the **continuity of the policy enforcement** to deal with changes in the access context.

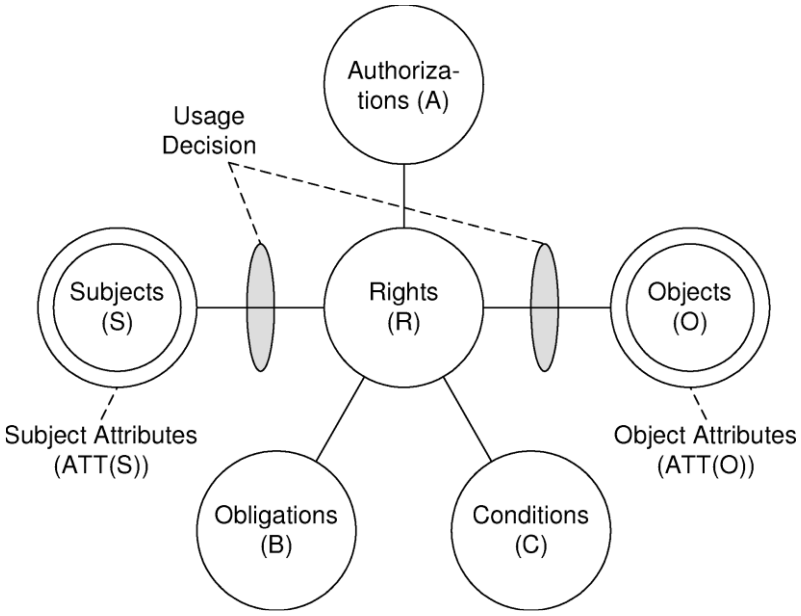

Figure 9 Usage Control Model [PS2004]

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 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. 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 E-CORRIDOR pilot. 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 10, 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.

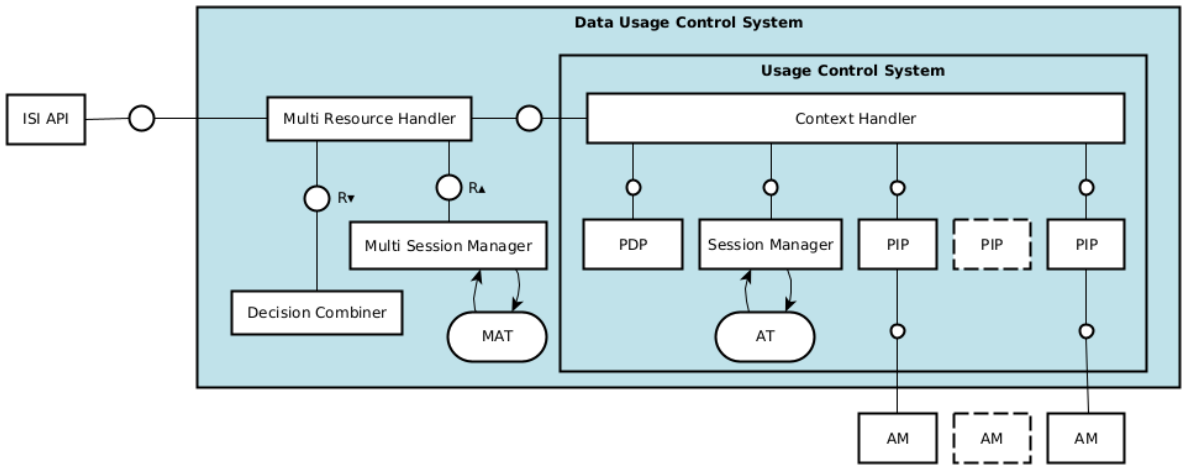

Figure 10 Data Usage Control System internal architecture

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, $d_1,...d_n$. 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 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 an 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 the by 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 the components in charge of managing the attributes that are required to evaluate the DSAs. For instance, an Identity Provider (IdP) could be adopted as AM to provide attributes such as the user nationality or age. These components should provide an interface to retrieve the current values of the attributes, that will be exploited by the Policy Information Points to interact with them. Some of them could also provide an interface to update attribute values. At the time of writing the E-CORRIDOR pilots identified the following pilot specific attributes:

Airport and Train Pilot:

- User Attributes
  a. Role
  b. Organization
  c. Business Sector
  d. Nationality
  e. Physical position
  f. Environment Attributes
  g. Number of people in the environment (e.g., in a waiting room)
  h. Emergency state (emergency, critical, normal)
  i. E-CORRIDOR framework integrity
- Data Attributes
  a. Type
  b. Producer Entity
  c. Producer Appliance
  d. Producer Appliance Owner
  e. Producer Appliance Physical Position
  f. Validity (binary value: if 0 the data cannot be used any more)

Smart City and Car Sharing pilot:

- User Attributes
  a. Role
  b. Organization
  c. Business Sector
  d. Private or Public service
  e. Nationality
  f. Physical position
  g. Environment Attributes
  h. E-CORRIDOR framework integrity
- Data Attributes
  a. Type
  b. Producer Entity
  c. Producer Appliance Owner
  d. Producer Appliance Physical Position
  e. Validity
  f. Read only entity

Information Sharing and Analytics Centre pilot

- User Attributes
  a. Role
  b. Organization
  c. Business Sector
  d. Nationality
  e. Environment Attributes
  f. Emergency state (emergency, critical, normal)
  g. E-CORRIDOR framework integrity
- Data Attributes
  a. Type
  b. Producer Entity
  c. Producer Appliance Owner
  d. Producer Appliance Physical Position

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, 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 AMs proposed by the Pilots, the following PIPs will be necessary:

- LDAP PIP, in charge of retrieving user attributes from an LDAP service;
- MySQL PIP, in charge of retrieving user attributes from a MYSQL service;
- Metadata PIP, in charge of retrieving data attributes from the metadata paired with Data Bundles.

Moreover, a further PIP which interacts with the DLI subsystem is integrated by default in the Usage Control System. This PIP 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.

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 on the data embedded in the Data Bundle (i.e., the Raw Data in the Data Protected Object in Figure 11) before making them available for the execution of the analytics or to the users for downloading. The DSA could require that different DMOs are executed on the same raw data depending on the 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 for the execution of an analytic.

Since each pilot could have its own specific DMOs, which depend on its privacy requirements and on the type of the data involved in the analytics defined by the pilot (a list of the data types taken into account by the E-CORRIDOR pilots is shown in Table 3 of deliverable D5.1), the DMO Toolbox has been designed to be able to easily integrate distinct operations that work on distinct data types. Hence, this component 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 using the required technology. From the technical point of view, the component will embed:

- native DMOs, i.e., DMOs embedded by the E-CORRIDOR framework, that are implemented as Java libraries that are deployed with the E-CORRIDOR framework code and that are directly invoked by the executeDMO function. The name of such DMOs and the corresponding Java library functions to be invoked are known at programming time;
- additional DMOs, that are integrated in the E-CORRIDOR framework at deployment time. These DMOs are implemented as RESTful services running within containers, which are deployed on the E-CORRIDOR server. The names of the DMOs that are reported in the policy and the corresponding service invocation are listed in a configuration file which is uploaded by DMO Toolbox component at startup time in the DMO table. Obviously, the same DMO names should have been used in the DSA vocabulary to allow policy makers to request the execution of such DMOs in the DSA. The DMO table is used by the DMO Toolbox to determine the service to be invoked to execute the DMO every time the executeDMO function is invoked and the DMO to be executed is not among the native ones.

At the time of writing, the DMOs required by the E-CORRIDOR pilots are the following:

- Airport and Train pilot: several sensitive data about the passengers are collected and managed including multi-biometrical data. Some data collected at reservation time (such as passenger name and age) and others in the airport and train station (such as passenger location and camera feeds) must be anonymized or pseudo-anonymized (e.g., with facial redaction in case of video streams). Moreover, information contained in the SIEM (Security Information and Event Management) logs like IP addresses and identifiers of the hardware devices should be anonymized before being transmitted to the ISAC pilot for cyber-security analysis. This anonymization function should preserve the data analytic capability of performing correlation analysis while avoiding to expose the IP address to parties external to the pilot. Other data such as passport number and boarding pass, network logs and models (e.g., for face recognition, passenger gait and behavior) being (highly) sensitive must be encrypted.

- Smart City and Car Sharing pilot: in the S2C pilot scenario, the eWallet data shared between the different mobility service providers with the purpose of registering and authenticating the users will not be manipulated. However, the eWallet data shared with the mobility consultancy (the micro-subsidies analytics toolkit) and with the itinerary-planning and carbon footprint calculation analytics needs to be pseudonymized to guarantee the privacy of travelers' personal data. In addition to pseudonymization, data for cyber-security analysis has to be obfuscated as it contains sensitive data (such as emails, logs, Car network data etc) that may allow the reidentification of the data owners and sources. Pseudonymization and data analysis on the Edge will guarantee a privacy-preserving driver behavioral recognition, obfuscation here is not a choice given the high accuracy needed for

effective data analysis results. Finally, the privacy preserving interest-based sharing will be based on Fully Homomorphic Encryption. This requires the execution of the encryption DMO, and it allows for the highest level of privacy thus enabling collaborative data sharing for sensitive personal data in the pilot.

- ISAC pilot: in the ISAC pilot scenario, multiple transportation organizations can share data concerning cyber-threats among companies, making them public. Such data can contain user personal and sensitive information that, actually, are not relevant to perform analytics on cyber-threats. For instance, the DMOs performed in the ISAC pilot are related to the anonymization of GPS information that are present in logs data provided by the transportation sector, or the identification numbers of vehicles.

4.4. Obligations Toolbox

The Obligations toolbox is the component of the ISI subsystem in charge of the execution of Obligations when required by the DSAs. Obligations are actions that must be executed by the DUCS and that do not involve the data embedded in the Data Bundles. The internal design of the Obligations Toolbox is the same as the one of the Data Manipulation Toolbox. Hence, the code of a number of native Obligations will be embedded in the Obligation Toolbox component, while additional obligations will be deployed as RESTful services and integrated in the E-CORRIDOR framework through a configuration file. As for the DMOs, also the Obligations that must be executed on each piece of data are determined by the DUCS 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.

At the time of writing, the Obligations required by the E-CORRIDOR pilots are the following:

- AT Pilot: as several highly sensitive data about the passengers are managed, to comply with the GDPR (General Data Protection Regulation) and local legislations, other than collecting only the minimum amount of information, data must be disposed properly once services and analysis have completed their operations and purposes. Therefore, appropriate data retention policy must be followed and the data must be removed: (i) at a specific date (e.g., once a trip is completed), (ii) after a given time (e.g., in case there is the need to allow the audit from the authorities) or once the analytics completes its analysis.

- S2C Pilot: to comply with legal obligations, in particular the right of service users (travelers) to rectify, modify or erase their data (the right to be forgotten), a mandatory obligation is needed to erase or modify data when requested. Another obligation is required to send emails to all partners who have or had access to a

specific piece of data in the past. Finally, data collected for Cyberthreat analysis must be deleted when the expiring date is reached.

- ISAC pilot: cyber-threat notification. In the ISAC pilot scenario, detecting new threats and vulnerabilities is the primary objective to prevent or mitigate attacks. The pilot offers intrusion and cyber-threat detection analytics analyzing data provided by the transportation sector organizations. After the analysis, the results should be notified by email or other communication systems for rapid awareness and timely implement the mitigation.

4.5. Bundle Manager

The Bundle Manager component of the Information Sharing Infrastructure subsystem is in charge of performing two different operations. During the packaging operation, the Bundle Manager creates a 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).

Figure 11 illustrates the Data Bundle structure.

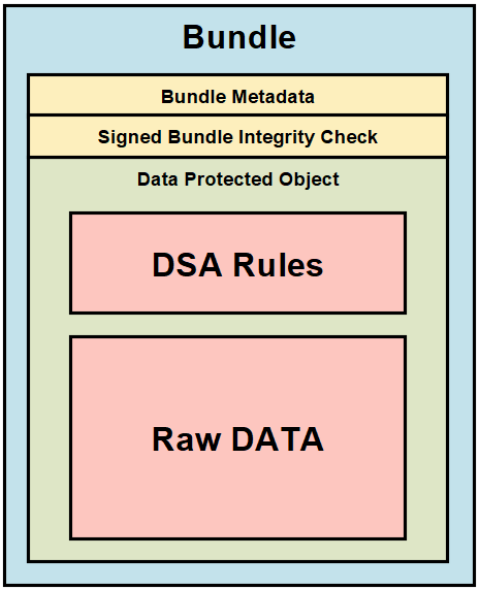

Figure 11 Data Bundle Structure

The complete description of the internal structure of the Data Bundle is given in 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. Moreover, user can decide whether the systems must encrypt uploaded data or not by specifying this in the corresponding DSA. Furthermore, in the context

of the E-CORRIDOR, the Bundle Manager should support different encryption models depending on 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 owners in the form of the data bundle. Each data bundle includes raw data, corresponding DSA and other relevant information. D5.2 better describes the implementation of the data bundle.

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). 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.

4.6.1 Bundle Store API

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. 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 API provides the following functionalities:

- CreateBundle: 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;
- ReadBundle: allows entities to retrieve the data bundle from the Bundle Store repository according to the given ID;
- DeleteBundle: removes the data bundle from the repository according to the corresponding ID;
- SearchBundle: 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 4 Data Bundle MetaData Example

Data Bundle Metadata Example


```
{
        "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"
}
```


Table 4 provides an example of the data bundle metadata described through attributes. In particular, the ID uniquely identifies the event, 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.

4.7. Buffer Manager

The ISI subsystem uses a Buffer Manager component, which is in charge of managing multiple data storage areas, known as Virtual Data Lakes (VDL). These Data Lakes are transient, i.e., they are created and reserved only for the execution of a specific analytic.

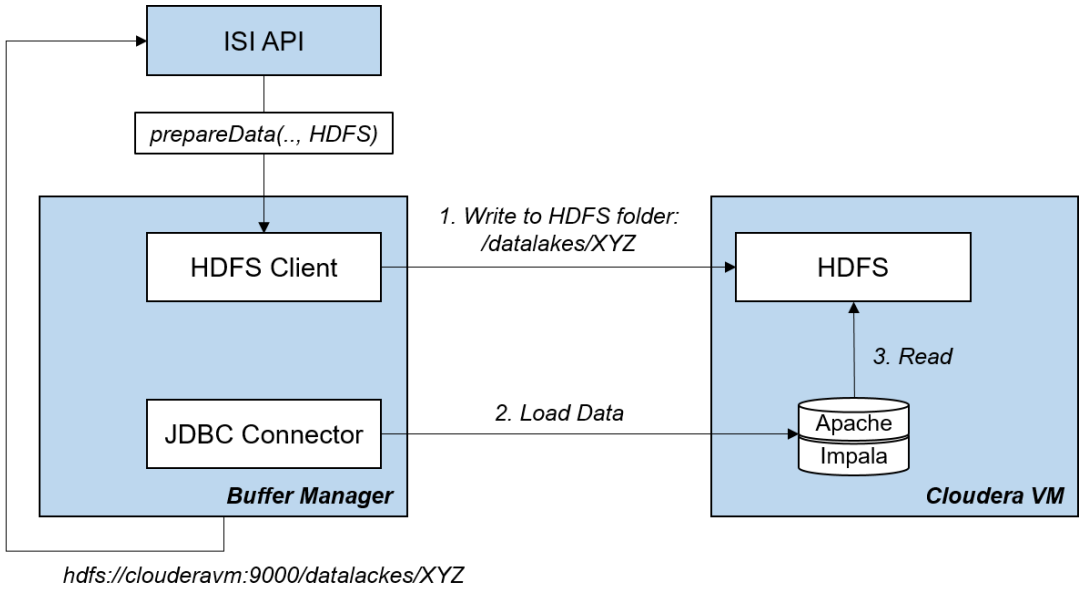

Figure 12 Write data to HDFS

Figure 12 shows the workflow between the Buffer Manager and the Apache Impala that runs on a Cloudera[2] 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.

[2] https://www.cloudera.com/products/open-source/apache-hadoop/impala.html

5. Information Analytics Infrastructure

The Information Analytics Infrastructure is the third subsystem of the E-CORRIDOR framework that is described in this deliverable. This component is devoted to the execution of the analytics that are required by the pilots on the data that have been stored in the E-CORRIDOR framework.

For instance, the ISAC pilot exploits the cyber-threat notification analytic (see Figure 13) to provide a system able to inform each transportation sector about new threats, vulnerabilities publicly discovered, and the related mitigation actions to actuate. The analytic is a notification service where a subscriber, i.e., a transportation enterprise or a vehicle, transmits a list of software and hardware applications in use and receives the list of vulnerabilities, correlated attack patterns, and mitigation for each application in the list. When a new threat or vulnerability is discovered, the subscriber is promptly notified.

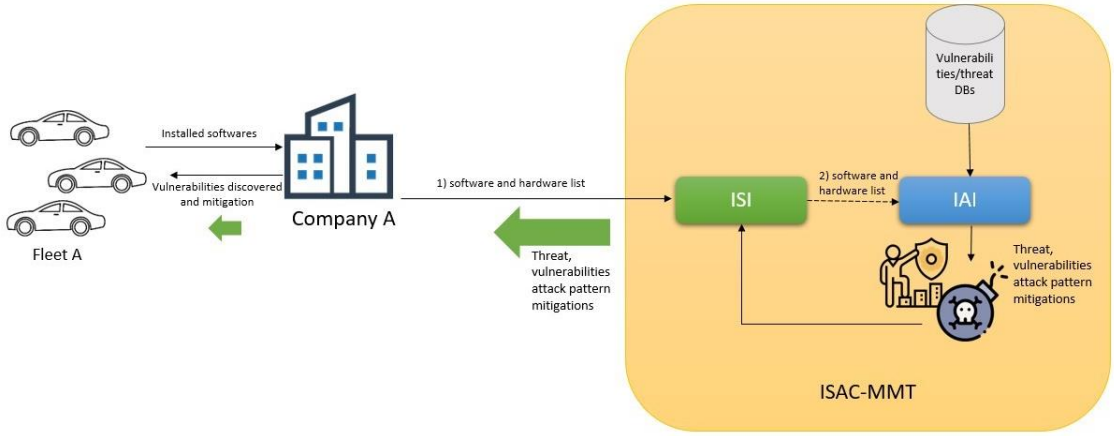

Figure 13 Cyber-threat notification analytic.

Figure 14 shows the internal architecture of the IAI subsystem.

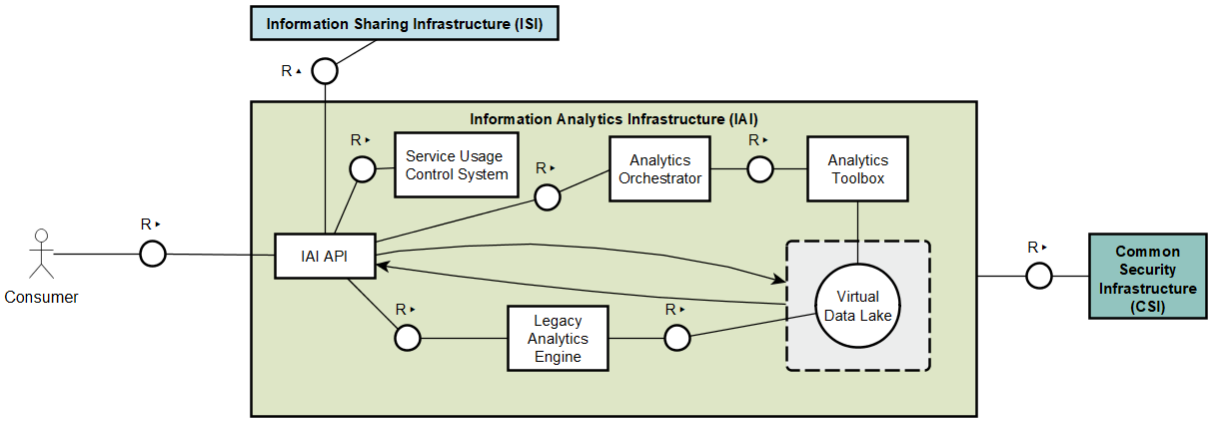

Figure 14 Information Analytics Infrastructure internal architecture

The users of the pilots of the E-CORRIDOR framework, using the pilot specific graphical interface (not shown in Figure 14) or directly using the APIs provided by the IAI subsystem (described in Section 5.1), selects a set of data and asks the execution of one of the provided analytic functions on them. The data selection is performed by imposing filters on the metadata paired with the data (e.g., all the data of type X produced from day D to day E). Once the analytic execution request has been submitted, the IAI returns a ticket to the user, and such ticket will be used to access the results of the analytic, which will be stored on the ISI subsystem when they will be available. The access to the results will be executed, again, either using the pilot specific graphical interface or using the APIs provided by the ISI subsystem.

The data that are used to run the analytic are stored on the ISI subsystem, and for this reason the IAI subsystem interacts with the ISI subsystem in order to collect the set of data requested by the user. As detailed in Section 4, the ISI subsystem, before making the requested data available for the execution of the analytic, enforces the DSA paired with each piece of data selected in order to make available for the analytic execution only the data whose DSA allows it, and also to manipulate the data for preserving their privacy before the analytics execution. In order to share such data with the IAI subsystem, the ISI subsystem creates a temporary Virtual Data Lake (see Section 4.7) which includes all the data available for the execution of the analytics and makes it available to the IAI subsystem. To enhance the data protection level, even this 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.

Finally, the IAI subsystem should be able to support the execution of a large and even growing set of analytics, according to the possibly dynamic requirements of the E-CORRIDOR pilots. To this aim, the IAI has been designed with the aim of easily allow the integration of analytics. The main idea is that further analytics could be added to IAI during its lifecycle with a minimal effort for their integration. In particular, some native analytics will be implemented as java libraries integrated within the IAI code, while other analytics will be implemented as RESTful microservices embed in containers.

The Sequence Diagram showing the complete workflow of the data analytics execution, which involves all subsystems of the E-CORRIDOR framework, is shown in Section 9 of deliverable D5.2.

5.1. IAI API

The IAI API component is the frontend of the IAI, and it exposes to the E-CORRIDOR users the interfaces to invoke the analytics provided by the E-CORRIDOR framework on a set of Data Bundles. As a matter of fact, Figure 14 shows that this component directly interacts with the E-CORRIDOR user, i.e., the consumer of the data. From the technical point of view, the IAI API component provides a RESTful interface where each analytic has its own API to be invoked.

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 analytic in such access context.

If the analytics execution is allowed, the IAI API continues the execution workflow by interacting with the ISI subsystem through the ISI API in order to trigger the creation of a Virtual Data Lake which includes the data available for the execution of the requested analytic function, possibly manipulated (e.g., anonymized) through the execution of the DMOs according to the related DSAs.

When the Virtual Data Lake is available, the IAI API component invokes the Analytics Orchestrator (described in Section 5.3) which coordinates the actual execution of the analytic functions on the data included in the Virtual Data Lake. As a matter of fact, each analytic exposed in the API could be a core analytic or a composition of existing ones. In the latter case, the Orchestrator component is in charge of managing the execution workflow.

When the result of the analytics is ready, the IAI API will invoke again the ISI subsystem through the ISI API in order to trigger the creation of a new Data Bundle including such result.

Finally, 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.

5.2. Service Usage Control System

The Service Usage Control System (SUCS) is aimed at regulating the usage of the IAI subsystem, i.e., at regulating the execution of the analytics by the E-CORRIDOR users, according to the model defined in [CDLMM2016]. This component provides a further protection of the E-CORRIDOR framework with respect to the DUCS. As a matter of fact, the difference between the DUCS that is embedded in the ISI subsystem and the SUCS is that the former protects the data according to the policies (DSA) defined by the data producers, while the latter protects the infrastructure by 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 IAI subsystem 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 DSA paired to them.

The policy enforced by the SUCS is written in UPOL. This policy is defined by the services administrators who directly upload them on the SUCS.

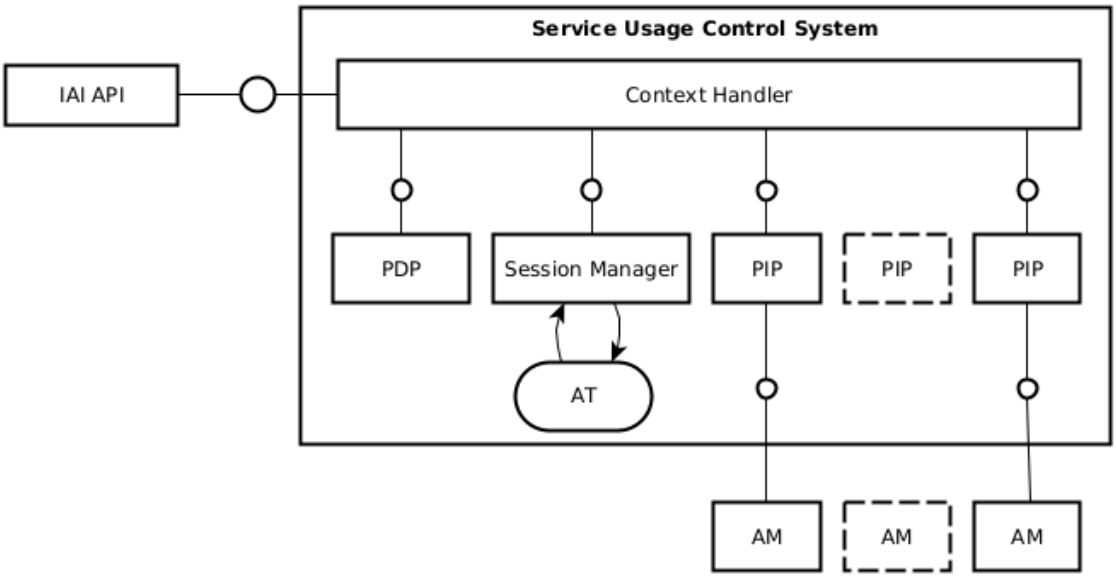

Figure 15 Service Level Usage Control System Internal Architecture

For what concerns the architecture, the SUCS component is simply another instance of the Usage Control System described in Section 4.2.1, as shown in Figure 15. Since the users who access the IAI are obviously the users who access the Data Bundles, the user attributes that are used in the service level usage control policies are the same that are used in the DSA (a list is available in Section 4.2.1) and, consequently, the PIPs and the AMs that are used in the SUCS are exactly the same that are used in the DUCS. 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 SUCS component 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

The analytics orchestrator aims at performing composition of analytics and workflow definition and management. The building blocks used by the orchestrator are the data analytics components available in the analytics toolbox of the IAI subsystem.

By following this approach, new and more complex services can be defined thanks to analytics orchestrator as the composition of the original ones with minimal effort on the development

side. Indeed, this requirement stem from the E-CORRIDOR project pilots by considering the complexity of some of the use cases that were defined in the previous project milestone. By adopting the analytics orchestrator the data analytics components can be kept simpler still allowing the E-CORRIDOR framework to provide complex services through a *modular* approach. All in all, the independent evolution of the single data analytics components is preserved, composed analytics can be offered to prosumers, and the instantiation of monolithic services is avoided. Actually, out of the IAI subsystem, even the E-CORRIDOR framework itself detects and treats the orchestrated analytics as any other data analysis component natively available in the analytics toolbox.

According to the requirements collected at the time of writing this deliverable, the composed workflows are specified at development or deployment time of the E-CORRIDOR framework. Therefore, the service compositions are *static* (i.e., cannot be changed during operations) and saved in the IAI subsystem.

The analytics orchestrator needs to provide:

a. a simple Domain Specific Language (DSL) (e.g., WS-BPEL [WSBPERL], Jolie [JOLIE]) to specify the workflow, either as a script (e.g., in YAML format) or with a simple code snippet written in a high level programming language (e.g., in Python);
b. the composition of the analytics in the toolbox through parallel and/or series interactions;
c. the support for monitoring the execution of the orchestrated services (mainly based on containers) and promptly reacting in case of problems for re-establishing the functioning of the workflow;
d. capabilities to be executed also on edge nodes with resource-constrained devices (as pilots consider the deployment of the E-CORRIDOR framework in smartphones and single board computers);
e. a wrapper to expose for the orchestrated analytics the same set of API each data analytics in the toolbox must comply with.

At the time of writing this deliverables, several off-the-shelf tools are under evaluation i.e., Puppet [PUPPET], Chef [CHEF], Ansible [ANSIBLE], SaltStack [SALT], Nornir [NORNIR].

An example of a YAML configuration file listing the steps in a prototypical workflow is presented in Figure 16.

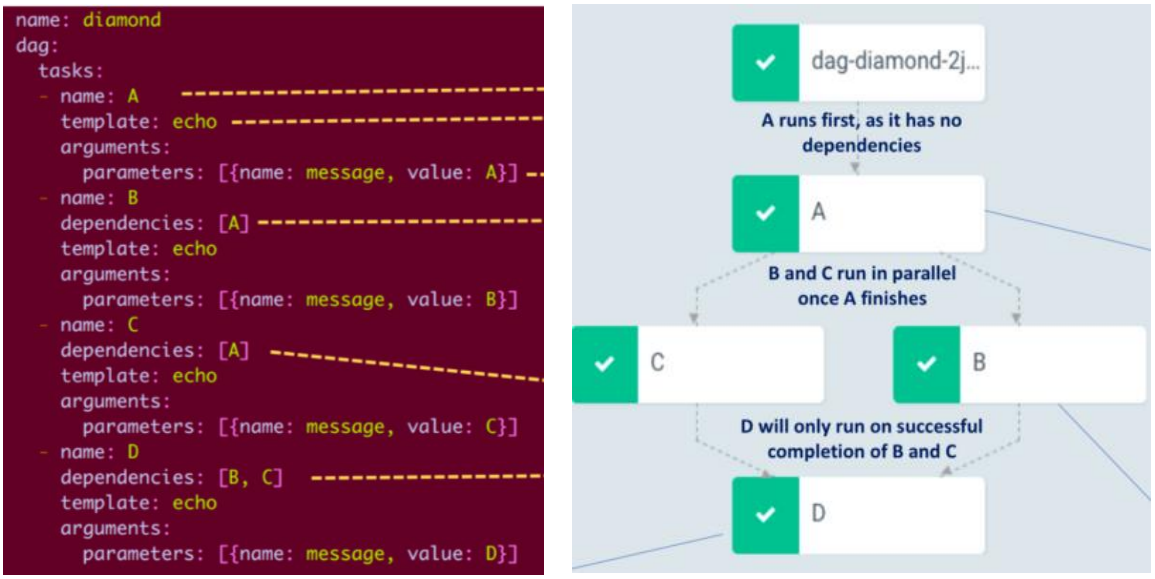

Figure 16 Example of workflow specified in YAML and its logical representation [ARGOK8]

5.4. Analytics Toolbox

The analytics toolbox is constituted by the set of data processing components available in the E-CORRIDOR framework with the aim of extracting knowledge from the data shared by the data producer. Data processed by the analytics in the toolbox include, but are not limited to: camera feeds, CANbus (controller area network) messages, OBD (on-board diagnostics) readings, IMU (inertial measurement unit) sensor data, RSSI (received signal strength indicator), travel preferences and directions. These tools are based on both machine learning and big-data analytics and can generally run in a hybrid edge-cloud fashion (specific restriction could be imposed by the technology providers e.g., in case of specific hardware requirements). Example of the analytics included in the first iteration of the E-CORRIDOR framework include secure routine for driver-identification, passenger location, trip planning, and intrusion detection systems.

Any pilot actor or external authenticated source can act as data producer, while the data consumer (the one who runs the analytics in this case) could be either different or the same (in the latter case it will act as *prosumer*).

As a matter of fact this toolbox is flexible and can accommodate new analytics (developed either by the pilots or by the technology providers of the E-CORRIDOR project) by following a *plugin* approach. This methodology will ease the addition of new analytics in case of raising needs from the pilots, simplify the deployment and protect the source code developed by the technology providers. Analytics are deployed in the IAI either as executable Java (JAR) or as service packaged in a software container (e.g., through Docker [DOC2021]) and can be added

dynamically (i.e., no change to the infrastructure is required to include new analytics to the toolbox).

Once an analytic registers itself with the orchestrator, the workflow involving the analytics toolbox is made by few steps:

1. The analytics orchestrator calls the requested analytics from the toolbox,
2. The analytics works on the virtual data lake created by the DMO toolbox,
3. Once the analytic is completed, the analytic orchestrator is informed and the data in output are saved as Data Bundles through the ISI API.

Data prepared by the Bundle Manager in the data lake can be accessed by the analytics as references to the Hadoop Distributed File System (HDFS) [HDFS2006] or to noSQL databases.

To be included in the toolbox each component just needs to expose an interface constituted by a very basic set of API:

- **START:** to start the data analysis over a given set of data made available by the Bundle Manager in the "virtual data lake";
- **STOP:** to gracefully stop the analytics and halt its execution (partial results are provided if available);
- **KILL:** to interrupt the analytic execution e.g., in case a policy violation is detected (no results are provided);

Once the analytic completes its execution, it needs to send an ***END*** message to inform the orchestrator that the analysis is complete, and that the data output is available and stored in the ISI.

Components in the analytics toolbox are managed as RESTful services and expose their interface by following the OpenAPI specification [OpenA2020]. In such a way, the analytic orchestrator can understand and consume services without requiring any knowledge on the service implementation and without any need to access to the code. The Swagger tool [SWA2001] is able to describe such OpenAPI in JSON format.

For the sake of clarity, the analytics in the toolbox, developed in this first maturation of the E-CORRIDOR platform, have been grouped as follows:

- Data analytics for driver and passenger identification
  - Secure Routine – driver identification
  - Passenger: Identification, Behavior, Context
  - Gait analysis – passenger authentication
  - Passenger location
- Privacy preserving itinerary planning
  - Multi-modal itinerary planning
- Privacy preserving (Security) analytics
  - Secure Multiparty-computation for Routine based authentication - Private Secure Routine
  - FHE – based checker
- Carbon foot print analytics
  - Carbon footprint analysis
- Intrusion detection technologies (IDS)
  - CAN bus IPS – EARNESTAutomotive Intrusion Detection
  - FHE - based intrusion detection

The analytics listed above have a Technology Readiness Level (TRL) spanning from level 3 to 4 (the research shows that the tool is feasible) and, thanks to the evaluation carried out in the project pilots, we expect to maturate these over the duration of the E-CORRIDOR project up to TRL 6 (the tool demonstrated its capability in a realistic environment). It is worth to note that the above set of analytics have been selected by taking into account the input received by the project pilots at the time of writing this document. But thanks to the plugin approach new analytics may be added in the toolbox to accommodate any potential additional need raised during the execution of the E-CORRIDOR project

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 analytic, 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.

6. Requirement Analysis

Deliverable D5.1 defined a set of requirements related to the E-CORRIDOR framework, which have been derived from the requirements defined by the three E-CORRIDOR pilots and which will allow such pilots to successfully exploit the E-CORRIDOR framework. This section examines such requirements in order to determine whether and how the architecture we defined in the previous sections fulfils them. This section follows the requirements organization defined in D5.1. Hence, requirements are divided in two major classes: functional and non-functional requirements.

6.1. Functional Requirements

Deliverable D5.1 classifies functional requirements in three sets:

- **Data Sharing**: Relates to the capabilities of sharing data between prosumers through the usage of DSAs and associated policies.
- **Data Analytics**: Concerns the possibility to run analytics services that use the shared data and produce a result, where both actions must obey to the applicable DSAs.
- **Data Manipulation Operations**: Describes a set of operations that can be applied to both the shared data and the execution of analytics services as well as their results, in order to transform or manipulate (fields of) the data to address specific needs, including privacy requirements or trustworthiness.

6.1.1. Data Sharing Requirements Analysis

This section reports the table of requirements defined in D5.1, concerning the capabilities required for the Information Sharing Infrastructure, and analyses each of such requirements to check whether it is satisfied by the E-CORRIDOR architecture previously defined.

Table 5 – Data Sharing Requirements Analysis

ID	Priority	Requirement Description	MET	Implementation
E-CORRIDOR-DS-01	MUST	The E-CORRIDOR framework provides a way to define **Data Sharing Agreements** between parties.	Y	The DLI subsystem allows to define DSAs through a user friendly graphical interface. The ISI subsystem allows to pair a DSA to each Data Bundle that is created and to enforce such DSA when the related Data Bundle is downloaded or used for analytics.
E-CORRIDOR-DS-02	MUST	E-CORRIDOR allows defining **multi-lateral DSAs**,	Y	The DLI subsystem allows to define the parties involved in a DSA when the DSA itself is created.

		i.e., between multiple parties (more than two).		
E-CORRIDOR-DS-03	MUST	E-CORRIDOR allows defining **policies** as a set of rules that regulate the data **sharing process** expressed in the DSA.	Y	The DLI subsystem allows to define DSA in terms of rules that regulate the data sharing, and the ISI subsystem allows to enforce the rules embedded in the DSA paired with a Data Bundle when such data object is used for analytics.
E-CORRIDOR-DS-04	MUST	E-CORRIDOR policies in the DSAs allow specifying conditions for **authorizations** (CAN), **obligations** (MUST), and **prohibition** (MUST NOT) logics.	Y	The DLI subsystem allows to define three kind of rules in a DSA: authorization rules, obligation rules, and prohibition rules. The ISI subsystem has been designed to properly enforce these three kinds of rules
E-CORRIDOR-DS-05	MUST	E-CORRIDOR allows sharing data (the "object" protected by the DSA) of **different formats**.	Y	The ISI subsystem allows to create and share Data Bundles embedding data having different formats. The specific format of the data embedded in a Data Bundle will be specified in a specific metadata field.
E-CORRIDOR-DS-06	MUST	E-CORRIDOR policies consider **end-user properties** within their logic.	Y	The DLI subsystem allows to define authorization/obligation/prohibition conditions that take into account end-user properties, called user attributes. The ISI subsystem has been designed to collect the current value of the user attributes required by a policy every time such policy must be evaluated and enforced.
E-CORRIDOR-DS-07	MUST	E-CORRIDOR policies consider **context**/**environment** properties within their logic.	Y	The DLI subsystem allows to define authorization/obligation/prohibition conditions that take into account context/environment properties, called environment attributes. The ISI subsystem has been designed to collect the current value of the environment attributes in a policy when a every time such policy is evaluated and enforced.

E-CORRIDOR-DS-08	MUST	E-CORRIDOR policies consider **time-based conditions** within their logic.	Y	The DLI subsystem allows to define time based conditions, and the ISI subsystem has been designed to properly trigger their evaluation and enforcement.
E-CORRIDOR-DS-09	MUST	E-CORRIDOR policies enable to express conditions to preserve **confidentiality** over the shared data.	Y	The DLI subsystem allows to define conditions to preserve the confidentiality of the data objects, and the ISI subsystem has been designed to properly enforce them by allowing the usage of the data object only if the requesting subject holds the proper rights.
E-CORRIDOR-DS-10	MUST	E-CORRIDOR policies enable to express **retention criteria** over the shared data (e.g., shared data is deleted after a certain amount of time or at a fixed date). (GDPR Requirement)	Y	The DLI subsystem allows to define retention criteria, and the ISI subsystem has been designed to properly trigger their evaluation and to enforce them.
E-CORRIDOR-DS-11	MUST	E-CORRIDOR policies enable to express conditions to control **access** to the shared data (i.e., conditions to evaluate before obtaining the data).	Y	The DLI subsystem allows to define authorization/obligation/prohibition conditions that control the access to the data objects, and the ISI subsystem has been designed evaluate these conditions before granting the access to the data objects.
E-CORRIDOR-DS-12	MUST	E-CORRIDOR policies enable to express conditions to control **usage** of the shared data (i.e., continuous authorisation after granted access).	Y	The DLI subsystem allows to define authorization/obligation/prohibition conditions that control the usage of the data objects, and the ISI subsystem has been designed to evaluate these conditions after the access has been granted and it is in progress, in order to take proper countermeasures in case of policy violation.
E-CORRIDOR-DS-13	MUST	E-CORRIDOR allows writing DSA policies for activating "**pre-processing rules**" on shared data, which are data	Y	The DLI subsystem allows to define obligations which specify data manipulation operations to be executed before the data object is used for the executing the analytics. The ISI subsystem has been

		manipulation operations (DMOs) performed before data is shared with other prosumers.		designed to execute such DMO when requested by the DSA paired with the Data Bundle which is being used.
E-CORRIDOR-DS-14	MUST	E-CORRIDOR data object preserves **ownership** of its stakeholder (i.e., the object data creator or the data collecting entity).	Y	The ISI subsystem, when creates a new Data Bundle, specifies in the data producer entity field the subject who requested the object creation
E-CORRIDOR-DS-15	MUST	E-CORRIDOR provides **DSA templates** to be used as pre-established or ad-hoc agreements to be instantiated by parties into DSAs.	Y	The DSA Lifecycle Infrastructure subsystem allow to save DSA templates that can be subsequently used to create new DSAs
E-CORRIDOR-DS-16	MUST	E-CORRIDOR provides **dynamic data sharing user preferences** to be configured by the producer (end-user), in an opt-in/opt-out fashion at data creation time.	Y	The DSA Lifecycle Infrastructure subsystem allows to define authorization conditions that take into account dynamic end-user properties, called mutable user attributes. The ISI subsystem has been designed to continuously collect and check the current value of mutable user attributes to detect changes and to react properly.
E-CORRIDOR-DS-17	MUST	E-CORRIDOR allows data producers to **accept or reject the DSA**, enabling to show the **parties** that will have access to the data, for which **purpose** and how the consumer(s) can **operate on** the data. (GDPR Requirement)	Y	The data producers can accept or reject a DSA depending on the DSA content.
E-CORRIDOR-DS-18	MUST	E-CORRIDOR enables **showing** to the producer **the DSA** applied to the data.	Y	The DSA Lifecycle Infrastructure subsystem allows the data producer entity to visualize the DSAs applied to the data object

E-CORRIDOR-DS-19	MUST	E-CORRIDOR receives data (**push mechanism**) from external sources (in particular CTI).	Y	The ISI subsystem provides APIs that can be invoked by external programs or scripts to create new data objects, i.e., to push data in the E-CORRIDOR framework.
E-CORRIDOR-DS-20	MUST	E-CORRIDOR **pulls data** (in particular CTI) from external sources, with specified polling intervals.	Y	The ISI subsystem provides APIs that can be invoked by external programs or scripts to create new data objects. The E-CORRIDOR framework will define scripts to pull data from external sources and create new Data Bundles form such data.
E-CORRIDOR-DS-21	MUST	E-CORRIDOR allows **using services** (i.e., access to them by API) of the Information Sharing Infrastructure **based on policy constraints**.	Y	The usage of the services provided by the E-CORRIDOR framework is regulated by the Service Usage Control policy.
E-CORRIDOR-DS-22	SHOULD	E-CORRIDOR allows defining **notifications policies** in the DSA that send a notification event (e.g., e-mail to a specified user or SMS) when the analytics service **result** is **generated** or have a **specific value** (e.g., IDS found a malicious data).	Y	The DLI subsystem allows to define obligations which specify when a notification, e.g., email messages or SMSs, must be sent. The ISI subsystem has been designed to trigger such notifications when required by the policy.
E-CORRIDOR-DS-23	MUST	E-CORRIDOR allows defining **notification policies** at data **ingestion** time (create/upload) and at data **read** time.	Y	The DLI subsystem allows to define obligations which specify when notifications, e.g., email messages or SMSs, must be sent. The ISI subsystem has been designed to trigger such notifications when required by the policy.
E-CORRIDOR-DS-24	SHOULD	E-CORRIDOR allows defining **notification policies** that use the **result data or metadata** as notification body text.	Y	The DLI subsystem allows to define new obligations to implement customized notification messages. The ISI subsystem has been designed to trigger such

				notifications when required by the policy.
E-CORRIDOR-DS-25	MUST	E-CORRIDOR attaches the DSA policies to the **shared data** (**sticky policy approach**).	Y	The ISI subsystem embeds the DSA in the Data Bundle along with the data to be shared.
E-CORRIDOR-DS-26	MUST	E-CORRIDOR allows **searching the collected data** by meta data set on the shared data (e.g., by data owner, party/parties collecting the data, purpose of sharing, date of sharing, and type/class of data).	Y	The ISI subsystem allows to embed metadata within the Data Bundles. These metadata will be used to filter the data stored on the ISI subsystem through proper queries.
E-CORRIDOR-DS-27	MUST	E-CORRIDOR allows **searching the DSAs** that are available for use depending on specific DSA properties (e.g., by parties in the agreement).	Y	The ISI subsystem allows to filter the data stored on the Bundle Store through proper queries, also exploiting some predefined DSA properties.
E-CORRIDOR-DS-28	SHOULD	E-CORRIDOR allows **updating shared data** by an authorised party (e.g., producer updating the data content).	Y	The E-CORRIDOR framework can update shared data embedded in a Data Bundle by deleting the old copy and uploading a new version having the same Data Bundle ID.
E-CORRIDOR-DS-29	MUST	E-CORRIDOR keeps a **logbook** of when the **operations on data** took place (e.g., create or read).	Y	The E-CORRIDOR framework integrates a logging system which keeps trace of the operations performed on the shared data. This system is part of the Common Security Infrastructure subsystem, that is not covered by this deliverable.
E-CORRIDOR-DS-30	SHOULD	E-CORRIDOR allows **notifying parties** of the DSA that a data object has been updated (via a log message). Update is considered as a	Y	The DLI subsystem allows to define new obligations to implement customized notification messages. The ISI subsystem has been designed to trigger such notifications when required by the policy.

		delete followed by a create operation.		
E-CORRIDOR-DS-31	MUST	E-CORRIDOR allows **revoking the DSA** to deny the access to all the related data.	Y	The DLI subsystem allows to mark an existing DSA as revoked. The ISI subsystem always checks that the DSA paired with a Data Bundle is valid before evaluating and enforcing it. If the DSA has been revoked, the data in the related Data Bundles will not be accessible any more.

6.1.2. Data Analytics Requirements Analysis

The E-CORRIDOR framework will provide data analytics as a service to support the pilot's analysis needs, through the IAI subsystem. This section reports the table of requirements that has been defined in D5.1 concerning the capabilities required for the IAI subsystem, and analyses each of such requirements to check whether it is satisfied by the E-CORRIDOR architecture previously defined.

.

Table 6 – Data Analytics Requirements

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR-DA-01	MUST	E-CORRIDOR allows defining **policies**, as a set of rules that regulate the data **analytics execution** over the shared data (e.g., on who can run a specific analytic and under which conditions).	Y	The IAI subsystem embeds a component, the Service Usage Control System, which enforces service level policies, i.e., usage control policies regulating which users are allowed to execute the analytics.
E-CORRIDOR-DA-02	MUST	E-CORRIDOR provides a way to transform data into a **common data format** before collecting it (in this way analytics services will work on the same format regardless of the data source). The format is related to	Y	The E-CORRIDOR framework, for each type of data, allows the user to decide to convert it in the standard format supported by E-CORRIDOR before uploading it on the ISI subsystem.

		the class or type of data (see Table 3 of deliverable D5.1).		
E-CORRIDOR-DA-03	MUST	E-CORRIDOR policies enable to express data sharing conditions over the **analytics results**.	Y	The DLI Subsystem allows to define DSAs which also include the DSA which will be paired with the result of the analytics run on the data the DSA is paired to.
E-CORRIDOR-DA-04	MUST	E-CORRIDOR policies over the analytics results include conditions on the **stakeholder relevance** (e.g., distribute results to stakeholders according to access control rules), by means of access control rules defined in the DSA policies.	Y	The DSA Lifecycle Infrastructure Subsystem allows to define DSAs which also include the polity which will be paired as new DSA with the result of the analytics run on the data the DSA is paired to.
E-CORRIDOR-DA-05	MUST	E-CORRIDOR allows running analytics **over data shared** by different parties, **according to (i.e., satisfying) individual parties' policies** in the DSA.	Y	The IAI subsystem allows to invoke analytics on a group of Data Bundles belonging to distinct producers. The DSA of each Data Bundle includes a specific set of conditions on the other Data Bundles in this group. The result is that some of the Data Bundles must be evicted from the group to satisfy the DSAs of the others, hence allowing the analytic execution.
E-CORRIDOR-DA-06	MUST	E-CORRIDOR allows **sharing analytics results** to parties in the DSA (e.g. it allows reading an analytics result/outcome).	Y	The DLI Subsystem allows to define DSAs which also include the DSA which will be paired with the result of the analytics run on the data the DSA is paired to
E-CORRIDOR-DA-07	MUST	E-CORRIDOR allows writing DSA policies for activating "**post-processing rules**" on an analytics	Y	The DLI subsystem allows to define obligations which specify data manipulation operations to be executed on the

		operation result, which are data manipulation operations (DMOs) performed before delivering the result to the prosumer.		results after the execution of an analytics.
E-CORRIDOR-DA-08	MUST	E-CORRIDOR attaches the DSA policies to the **analytics result** data (**sticky policy approach**).	Y	The ISI subsystem creates a new Data Bundle to represent analytics results. Hence, a DSA is embedded in the Data Bundle.
E-CORRIDOR-DA-09	MUST	E-CORRIDOR allows **searching the generated analytics results data** (e.g., by owner/party, time, etc.).	Y	The analytic results will be stored on the ISI, and the ISI subsystem allows to filter the data stored on the ISI subsystem through proper queries.
E-CORRIDOR-DA-10	MUST	E-CORRIDOR provides **encrypted communication channels** to clients.	Y	The communications among the DLI subsystem, the ISI subsystem, the IAI subsystem and the users are on top of https channels.
E-CORRIDOR-DA-11	SHOULD	E-CORRIDOR allows using Fully Homomorphic Encryption (**FHE**) based analytics.	Y	The E-CORRIDOR framework supports the execution of FHE based analytics by properly defining the DMO operations executed on the related data before storing them on the ISI subsystem.

6.1.3. Data Manipulation Operations

As shown in Section 4.3, the E-CORRIDOR framework allows to define operations, called DMOs, that modify the data embedded in a Data Bundle before making them available for the execution of the analytics or for downloading. This section reports the table of requirements concerning the DMOs that has been defined in D5.1, and analyses each of such requirements to check whether it is satisfied by the E-CORRIDOR architecture previously defined.

Table 7 – Data Manipulation Requirements Analysis

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR-DM-01	MUST	E-CORRIDOR allows **data obfuscation** or **anonymization** of specific data fields (e.g., for biometrical user data). (GDPR Requirement)	Y	The ISI subsystem allows to easily integrate new Data Manipulation Operations that are required by the pilots, that will be invoked when required by the DSA. The vocabulary can be easily extended to allows the users to use such new DMOs in their DSA.
E-CORRIDOR-DM-02	MUST	E-CORRIDOR allows **data pseudo-anonymization** of specific data fields. (GDPR Requirement)	Y	As above
E-CORRIDOR-DM-03	MUST	E-CORRIDOR allows **data encryption** of specific data fields (e.g., for biometrical user data).	Y	As above
E-CORRIDOR-DM-04	SHOULD	E-CORRIDOR allows Fully Homomorphic Encryption (**FHE**) of data (email addresses or IP addresses).	Y	As above

6.2. Non-Functional Requirements

Deliverable D5.1 defines non-Functional Requirements (NFRs) and organizes them in the following sets:

- **Security**: Since E-CORRIDOR focuses strongly on security aspects, Information Technology security requirements, including privacy and the regulatory needs are of prime importance
- **Operational**: This set of requirements define specific characteristics of the environments where the pilots operate
- **Performance**: Pilots have requirements related to the impact on their operative flows, e.g., concerning the maximum amount of time users will wait for specific services. E-CORRIDOR analytics services and DMOs (including homomorphic computing and encryption) might have high memory and computation necessities.
- **Usability**: The E-CORRIDOR framework should provide services that are effective and easy to consume, as well as it should enable pilots to create such kind of services.

6.2.1. Security Requirements Analysis

This section analyses the requirements defined in D5.1 that address properties of **confidentiality**, **integrity,** and **availability** (CIA), as well as **authentication**, **authorisation**, **non-repudiation**, and **accountability**.

Table 8 – Analyses of the Security Requirements for Information Security

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR-Sec-IS-01	MUST	E-CORRIDOR traces with **audit trails** the DSA policies evaluations, like authorisations outcomes (e.g. grant or deny access to a shared data), analytics execution, and data manipulation operations, for auditing purposes including accountability, non-repudiation and compliance.	Y	The E-CORRIDOR framework integrates a logging system which keeps trace of the operations performed on the shared data. This system is part of the Common Security Infrastructure that is not covered by this deliverable

E-CORRIDOR-Sec-IS-02	MUST	E-CORRIDOR stores **data encrypted at-rest** to preserve confidentiality and privacy.	Y	The Data Bundles that are stored in the ISI subsystem are encrypted.
E-CORRIDOR-Sec-IS-03	MUST	E-CORRIDOR **protects data in-transit** (e.g., using TLS protocol) with **encrypted channels** to collect (e.g., upload) or deliver data (e.g., read), allowing to preserve confidentiality, privacy and authenticity.	Y	The communications among the DLI subsystem, the ISI subsystem, the IAI subsystem and the users are on top of https channels.
E-CORRIDOR-Sec-IS-04	MUST	E-CORRIDOR employs **data integrity** measure over the shared data.	Y	The Data Bundles that are stored in the ISI subsystem are signed.
E-CORRIDOR-Sec-IS-05	SHOULD	E-CORRIDOR uses capabilities to evaluate the **integrity** of the running framework.	Y	The E-CORRIDOR framework has an integrity evaluation procedure, and the current integrity status is stored in an environmental attribute that could be used in DSA.
E-CORRIDOR-Sec-IS-06	MUST	E-CORRIDOR provides its **API** functionalities after performing **authentication** and **authorisation** steps by using standard protocols (e.g., OpenID Connect, OAuth2).	Y	The API provided by the DLI subsystem, the ISI subsystem, the IAI subsystem require user authentication.

The next table analyses the **Regulatory/Compliance requirements**.

Table 9 – Analysis of Security Requirements for Regulatory/Compliance

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR-Sec-RC-01	MUST	E-CORRIDOR enables to define policies for **being compliant** with (some[3]) prescriptions of regulations (e.g. GDPR) and privacy needs. For example, these policies could allow expressing functional needs of access control or data anonymization, as those presented in Sections 6.1.1, 6.1.2, 6.1.3 of D5.1.	Y	All the requirements previously defined concerning GDPR are satisfied.

6.2.2. Operational Requirements Analysis

This section analyses the requirements defined in D5.1 concerning framework deployment, split between distributed computing, and extensibility and interoperability requirements.

Table 10 – Distributed Computing Requirements Analysis

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR Ope-01	MUST	E-CORRIDOR provides a distributed computing deployment model both at the **edge** and at the **cloud**.	Y	The subsystems of the E-CORRIDOR architecture could be deployed on several servers and they could be configured to interact among them.

[3] We understand that being compliant with a regulation is not only a matter of DSA policies, but here we focus on policies that can support some regulatory prescriptions (e.g., data pseudo-anonymization or data retention).

E-CORRIDOR Ope-02	MUST	E-CORRIDOR provides **analytics at the edge** capability.	Y	Customized versions of the IAI subsystem that can be deployed and executed on the edge will be defined
E-CORRIDOR Ope-03	MUST	E-CORRIDOR provides **collaborative analytics** at the **cloud**.	Y	The IAI subsystem allows to select set of Data Bundles produced by distinct data producer entities, and to execute collaborative analytics on these data sets.

In the next table we examine requirements defined in order to enable the integration with the E-CORRIDOR framework to build a system that is both easily extensible and interoperable.

Table 11 – Extensibility and Interoperability Requirements Analysis

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR Ope-04	MUST	E-CORRIDOR provides a **standard way** (e.g., guidelines, API, code skeleton template, etc.) for creating E-CORRIDOR **compliant analytics services**.	Y	The guidelines and the skeleton to develop E-CORRIDOR compliant analytics services will be provided.
E-CORRIDOR Ope-05	SHOULD	E-CORRIDOR provides an **asynchronous** way to run analytics services.	Y	The IAI API that allows to execute an analytic is asynchronous. The API returns a ticket that will be used to retrieve the results once they are ready.
E-CORRIDOR Ope-06	SHOULD	E-CORRIDOR provides a **standard way** (e.g., guidelines, API, code skeleton template, etc.) for **creating a DMO** (Data Manipulation Operations).	Y	The guidelines and the skeleton to develop E-CORRIDOR compliant Data Manipulation Operations will be provided.
E-CORRIDOR Ope-07	MUST	E-CORRIDOR provides its functionalities through an	Y	The E-CORRIDOR services are REST based.

		Application Programming Interface (**API**) based on the **REST** principle and open standards (e.g., OpenAPI 0).		
E-CORRIDOR Ope-08	MUST	E-CORRIDOR framework is delivered via the **micro-services** architectural pattern (based on **containers**).	Y	The E-CORRIDOR framework consists of a number of micro-services, some of which container based.

6.2.3. Performance Requirements Analysis

This section analyses the requirements defined in D5.1 concerning system performance.

Table 12 – Performance Requirements Analysis

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR Per-01	MUST	E-CORRIDOR reduces the **data transfer** between the edge and the cloud.	Y	The E-CORRIDOR framework allows to perform some analytics directly on the edge, in order to transfer to the cloud the result only, thus avoiding to transfer all the data to be processed.
E-CORRIDOR Per-02	MUST	E-CORRIDOR **authentication mechanism** should have a low performance impact.	Y	The E-CORRIDOR framework adopts single-sign on authentication in order to reduce the interactions with the users, thus minimizing the impact on performance.
E-CORRIDOR Per-03	SHOULD	E-CORRIDOR allows sharing (uploading) a **large amount of data**.	Y	The ISI subsystem allows to store a very large quantity of data which can used to perform analytics. The E-CORRIDOR framework allows to have several cooperating ISI subsystems, thus increasing the quantity of data that can be managed by the framework.

6.2.4. Usability Requirements Analysis

This section analyses the requirements defined in D5.1 concerning usability, i.e., the effectiveness and easiness of use of a solution as well as to be learned to become profitable in its execution with the least amount of time as possible.

Table 13 – Usability Requirements Analysis

ID	Priority	Requirement	MET	Implementation
E-CORRIDOR Use-01	MUST	E-CORRIDOR uses **standard authentication protocols** (e.g. OpenID Connect 0, OAuth2 0, SAML 0, eIDAS 0)	Y	The authentication will be based on eIDAS and SAML standards. Please see Deliverable D8.1 for more details.
E-CORRIDOR Use-02	SHOULD	E-CORRIDOR usage allow seamless authentication by leveraging **Single-Sign On** (SSO) authentication schema.	Y	The E-CORRIDOR framework adopts single-sign on authentication in order to reduce the interactions with the users. Please see Deliverable D8.1 for more details.

7. Contribution to the Achievement of the Project Objectives

The E-CORRIDOR framework aims at providing a collaborative, privacy-aware and edge-enabled platform for information sharing, data analysis and security services to multimodal transportation domains.

For a complete account on how the different infrastructures contribute to the objective the reader can refer to D5.2. Here we briefly recall the main objectives and present a table that focus on the specific contributions of the ISI and IAI.

Overall, E-CORRIDOR have the following objectives (and here reported for the sake of completeness):

- Objective 1: E-CORRIDOR will build a flexible, confidential and privacy-preserving framework for managing data sharing, for several purposes, by different prosumers.
- Objective 2: E-CORRIDOR will define edge enabled data analytics and prediction services in a collaborative, distributed and confidential way.
- Objective 3: E-CORRIDOR will define a secure and robust platform in holistic manner to keep the communication platform safe from cyber-attacks and ensure service continuity.
- Objective 4: E-CORRIDOR will improve, mature and integrate several existing tools provided by E-CORRIDOR partners and will tailor those to the specific needs of the E-CORRIDOR platform and Pilots.
- Objective 5: E-CORRIDOR will provide mechanisms for seamless access to multimodal transport.
- Objective 6: the framework and the services developed will be used to deliver three pilot products.
- Objective 7: E-CORRIDOR will be promoted and ease the exploitation, communication, standardization, dissemination and early adoption of its results.

The contributions of the E-CORRIDOR framework to the E-CORRIDOR objectives are as follow. Table 14 focuses on the ISI and IAI contributions to the project objectives.

Table 14 ISI and IAI contributions to the project objectives.

	Obj. 1	Obj. 2	Obj. 3	Obj. 4	Obj. 5	Obj. 6	Obj. 7
Deployment model: Edge-enabled architecture		✓	✓				
Information Sharing Infrastructure supporting Data Bundles	✓	✓		✓		✓	✓
DSA-based data sharing	✓	✓					✓

Flexible Information Analytics Infrastructure		✓	✓	✓	✓	✓	

8. Conclusion

This deliverable described the internal architecture and the functionality of the ISI and of the ISI subsystems, which provide two main functionalities at the core of the E-CORRIDOR framework, i.e., the privacy preserving data sharing and the data analytics execution. Moreover, this deliverable also covers the DLI subsystem, because it allows to create the DSAs that express the privacy preferences to be enforce on the Data Bundles uploaded on the E-CORRIDOR framework.

The previously mentioned subsystems are inherited from the EU H2020 funded project "Collaborative and Confidential Information Sharing and Analysis for Cyber Protection" (H2020-DS-2015-1 C3ISP – GA#700294), and this deliverable describes how their internal architectures and functionalities have been extended and maturated in order to accommodate the needs of the E-CORRIDOR pilots. Since such needs have been summarized by a number of framework requirements, described in deliverable D5.1, the maturation of the aforementioned subsystems started from such requirements. A relevant difference between the C3ISP and the E-CORRIDOR sharing platforms is that the former is meant for a specific type of data and analytics, the Cyber Threat Information, while the latter is a general data sharing and analysis platform, not designed for a specific kind of data. As a matter of fact, the E-CORRIDOR pilots consider a very wide set of distinct data types (described in Section 3.1.1 of deliverable D5.1), from data collected from the CAN BUS of vehicles to the passengers' passports. Consequently, also the kind of analysis performed by the two platforms are very different. Again, the E-CORRIDOR framework is meant to be very general, and to be able to integrated several kind of analytic functions operating on several kinds of data. For these reasons, a number of components of the E-CORRIDOR framework has been designed to be general. For instance, both the DMO and the Obligation toolboxes do not provide a predefined and built-in number of functionalities, but the DMOs and the obligations required by the use cases can be easily integrated in the E-CORRIDOR framework, provided that they are implemented as RESTful services. Similarly, the IAI subsystem includes two new components, Analytics Toolbox and the Analytics Orchestrator, which allow to easily integrate the analytics needed by each use case, and to compose such analytics to obtain new ones.

Finally, this deliverable confirms that the architecture defined for the ISI, IAI, and DLI subsystems contribute to satisfy the requirements defined for the E-CORRIDOR framework. In particular, a description of how the proposed subsystems contribute to satisfy each of the requirements Framework Requirements expressed in D5.1 is given in Section 6.

9. References

Here we provide bibliography references used in the document:

[AVH2003] Grigoris Antoniou and Frank Van Harmelen. Web Ontology Language: OWL. In Hand-book on Ontologies in Information Systems, pages 67–92. Springer, 2003.

[MPS2010] I. Matteucci, M. Petrocchi,and M.L. Sbodio. CNL4DSA: a Controlled Natural Language for Data Sharing Agreements. In *SAC: Privacy on the Web Track*. ACM, 2010. 21, 23, 52

[XACML2013] OASIS. eXtensible Access Control Markup Language (XACML) Version 3.0, January 2013. 21, 52, 53

[GLT1988] K. Guldstrand Larsen and B. Thomsen. A modal process logic. In *LICS*, pages 203–210, 1988. 23

[MPSW2011] I. Matteucci, M. Petrocchi, M.L. Sbodio, and L. Wiegand. A design phase for data sharing agreements. In *DPM/SETOP*, pages 25–41. Springer, 2011. 34, 52, 53

[PS2004] J. Park and R. Sandhu. $UCON_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur. 7 (1) (2004), 128–174

[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

[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

[BAL2021] WSO2 Balana implementation: https://github.com/wso2/balana (fetched April 2021)

[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

[WSBPEL] Web Service Business Execution Language: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[JOLIE] JOLIE: https://jolie-lang.org/

[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

[SALT] SaltStack: https://docs.saltproject.io/en/latest/topics/orchestrate/index.html

[NORNIR] Nornir: https://nornir.readthedocs.io/en/latest/plugins/execution_model.html

[ARGOK8] Argo Workfloe Engine for Kubernetes: https://itnext.io/argo-workflow-engine-for-kubernetes-7ae81eda1cc5

[OPENID2021] OpenID Connect, OpenID Foundation (2014), https://openid.net/connect/

[OAUTH2012] The OAuth 2.0 Authorization Framework - RFC6749, Internet Engineering Task Force (IETF), D. Hardt, Ed., Microsoft, October 2012, https://tools.ietf.org/html/rfc6749

[SAML2005] Security Assertion Markup Language (SAML), v2.0 (2005), Security Services (SAML) Technical Committee, https://wiki.oasis-open.org/security/FrontPage

[EIDAS2020] electronic IDentification, Authentication and trust Services (eIDAS), EU Regulation 910/2014, https://eur-lex.europa.eu/eli/reg/2014/910/oj

[OPENA2020] OpenAPI Specification, OpenAPI Initiative, 2020, http://spec.openapis.org/oas/v3.0.3

[SWA2001] Swagger, an Interface Description Language for describing RESTful APIs expressed using JSON, 2001, https://swagger.io/

[DOC2021] Docker, OS-level virtualization, 2013, https://www.docker.com/

[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