



D8.2

Components First Maturation

WP8 – C3ISP Data Sharing, Analytics and Crypto Technology Maturation

C3ISP

*Collaborative and Confidential Information Sharing and Analysis for Cyber
Protection*

Due date of deliverable: 30/09/2018
Actual submission date: 30/09/2018

26/09/2018

Version 1.0

*Responsible partner: CEA
Editor: Thanh Hai Nguyen
E-mail address: thanhhai.nguyen@cea.fr*

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)	



*The C3ISP Project is supported by funding under the Horizon 2020
Framework Program of the European Commission DS 2015-1, GA #700294*

Authors: T.H. Nguyen, V. Herbert (CEA), M. Manea, P. Ciampoli, M. Russo (HPE), I. Herwono (BT), R. de Lemos, D. Chadwick, J. Ziembicka, W. Fan (UNIKENT), F. Di Cerbo, J. Boehler (SAP), P. Mori, A. Saracino, G. Costantino, I. Matteucci (CNR), J. Dobos, C. Wong (3D REPO), R. Hamouda (GPS), S. Tranquillini, A. Arighi (CHINO)

Approved by: C. Wong, J. Dobos (3DREPO), P. Niamadio (GPS)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	20/07/2018	Paolo Mori / Thanh Hai Nguyen	CNR / CEA	Initial ToC
	02/08/2018	Ian Herwono	BT	Section 6 – new contents
0.2	22/08/2018	Jonas Böhler	SAP	Section 7 intro, 7.1, 7.2 – new content
	27/08/2018	Thanh Hai Nguyen	CEA	Section 5.10 – added Bundle Manager description
	30/08/2018	Paolo Mori	CNR	Section 5.7 – added component description, maturation status and requirements
	04/09/2018	Joanna Ziembicka	UNIKENT	Section 5.2 - added DPOS component description
	07/09/2018	Ilaria Matteucci	CNR	Section 4.2 – added DSA Mapper description, maturation, requirement status at M24 and First release description
0.3	07/09/2018	Gianpiero Costantino	CNR	Section 6.1 – added monitoring DGA analytic description, maturation, and First release description
	10/09/2018	Joanna Ziembicka	UNIKENT	Section 5.2 – added first release of the component description
0.4	11/09/2018	Ian Herwono	BT	Section 7.3, 7.4 – added requirement and source code description Section 9 – added full text
	12/09/2018	Jonas Böhler	SAP	Section 7 – new content for first release
0.5	12/09/2018	Vincent Herbert	CEA	Section 8.2 – added source code description
0.6	13/09/2018	Andrea Saracino	CNR	Section 6.1 – added email spam detection analytic description, maturation, and First release description
0.7	14/09/2018	Anrdrea Saracino, Antonio La Marra	CNR	Section 5.7 – added First release description
	19/09/2018	Paolo Mori / Andrea	CNR	Section 5.7, 6, 6.1
0.8	20/09/2018	Thanh Hai Nguyen	CEA	Merged version from partner contributions
	25/09/2018	Thanh Hai Nguyen	CEA	Merged version from internal reviewers 3DREPO and GPS, updated from HPE's, SAP's, CHINO's contributions

0.9	26/09/2018	Thanh Hai Nguyen	CEA	From CNR – add missing introduction text and references
1.0	26/09/2018	Thanh Hai Nguyen	CEA	Finish for integrating

1. Executive Summary

Deliverable 8.2 is the second output of Work Package 8, “C3ISP Data Sharing, Analytics and Crypto Technology Maturation” due at M24. This deliverable describes the first software releases of the components in the C3ISP framework: the source code description, a potential improvement for the next version and an accompanying document (this document).

Table of contents

1. Executive Summary	4
2. Introduction	8
2.1. Overview	8
2.2. Deliverable Structure	8
3. High-Level Architecture	9
4. Data Sharing Agreements.....	10
4.1. DSA Editor	10
4.1.1. Component description	10
4.1.2. Maturation status	15
4.1.3. Addressing specific privacy requirements in DSAs.....	23
4.1.4. Requirement Analysis at M24.....	23
4.1.5. First release of the component	25
4.2. DSA Mapper.....	31
4.2.1. Component description	31
4.2.2. Maturation status	32
4.2.3. Requirement Analysis at M24.....	33
4.2.4. First release of the component	34
5. Data collection and Usage Enforcement: The Information Sharing Infrastructure (ISI) .	37
5.1. Information Sharing Infrastructure API	37
5.1.1. Component description	37
5.1.2. Maturation status	37
5.1.3. Requirement Analysis at M24.....	37
5.1.4. First release of the component	38
5.2. Data Protected Object Storage (DPOS).....	42
5.2.1. Component description	42
5.2.2. Maturation status	42
5.2.3. Requirement Analysis at M24.....	43
5.2.4. First release of the component	43
5.3. Buffer –Manager.....	48
5.3.1. Component description	48
5.3.2. Maturation status	49
5.3.3. Requirement Analysis at M24.....	49
5.3.4. First release of the component	49
5.4. Format Adapter.....	53
5.4.1. Component description	53
5.4.2. Maturation status	53

- 5.4.3. Requirement Analysis at M24..... 54
- 5.4.4. First release of the component 54
- 5.5. DSA Adapter Frontend..... 56
 - 5.5.1. Component description 56
 - 5.5.2. Requirement Analysis at M24..... 57
 - 5.5.3. First release of the component 58
- 5.6. Event Handler 62
 - 5.6.1. Component description 62
 - 5.6.2. Maturation status 62
 - 5.6.3. Requirement Analysis at M24..... 62
 - 5.6.4. First release of the component 63
- 5.7. Continuous Authorization Engine 67
 - 5.7.1. Component description 67
 - 5.7.2. Maturation status 67
 - 5.7.3. Requirement Analysis at M24..... 69
 - 5.7.4. First release of the component 71
- 5.8. Obligation Engine..... 78
 - 5.8.1. Component description 78
 - 5.8.2. Maturation status 78
 - 5.8.3. Requirement Analysis at M24..... 79
 - 5.8.4. First release of the component 80
- 5.9. Data Manipulation Operation Engine..... 84
 - 5.9.1. Component description 84
 - 5.9.2. Maturation status 84
 - 5.9.3. Requirement Analysis at M24..... 85
 - 5.9.4. First release of the component 85
- 5.10. Bundle Manager..... 88
 - 5.10.1. Component description..... 88
 - 5.10.2. Maturation status 88
 - 5.10.3. First release of the component..... 89
- 6. Collaborative Data Analytics: The Information Analytics Infrastructure (IAI) 92
 - 6.1. Specific Data Analytics Examples 92
 - 6.1.1. Component description 92
 - 6.1.2. Maturation status 92
 - 6.1.3. Requirement Analysis at M24..... 93
 - 6.1.4. First release of the component 93
- 7. Visualization of Security Analytics..... 99

7.1.	Component description.....	99
7.2.	Maturation status	99
7.3.	Requirement Analysis at M24	100
7.4.	First release of the component.....	101
8.	Anonymization and Homomorphic Encryption Algorithms	102
8.1.	Anonymization Algorithms	102
8.1.1.	Component description	102
8.1.2.	Maturation status	102
8.1.3.	Requirement Analysis at M24.....	103
8.1.4.	First release of the component	104
8.2.	Homomorphic Encryption Algorithms	106
8.2.1.	Component description	106
8.2.2.	Maturation status	107
8.2.3.	Requirement Analysis at M24.....	108
8.2.4.	First release of the component	109
9.	Managed Security Services	114
9.1.	Component description.....	114
9.2.	Maturation status	115
9.3.	Requirement Analysis at M24	115
9.4.	First release of the component.....	115
10.	Conclusions	117
11.	References	118
Appendix 1.	DSA parent ontology for DSA Editor	120

2. Introduction

2.1. *Overview*

This deliverable is the second output of WP8, “C3ISP Data Sharing, Analytics and Crypto Technology Maturation” due at M24. WP8’s main goal is the maturation of a set of tools and technologies that are provided by the C3ISP partners and that can be exploited for the implementation of the Information Sharing Infrastructure (ISI) and of the Information Analysis Infrastructure (IAI) of the C3ISP Framework.

For each of the tools provided by the C3ISP partners, this document provides: i) a description of the new functions developed in the last year, explaining why they are needed in C3ISP, taking as reference the maturation we promised at Y1 (where possible); ii) a description of the first release of the component, including implementation details.

2.2. *Deliverable Structure*

This document is structured as follows. Section 2, for the convenience of the readers, reports a high-level view of the architecture of the C3ISP Framework. For a detailed description of the architecture, of its components and of the interactions among them please refer to D7.3. Each section from Section 3 to Section 8 covers one of the functionalities required in the C3ISP Framework and give the maturation status at M24 of the components using related tools provided by the C3ISP partner. Finally, Section 9 draws the conclusions.

3. High-Level Architecture

This section briefly recalls the new version of the high level C3ISP Framework reference architecture (shown in Figure 1) that has been defined at month 24. The main aim of this section is to give a quick overview of the main components of the architecture and a brief description of their main functionalities. A very detailed description of the components of the architecture, of their functionalities, of their interactions, and of the workflow of the main operations of the C3ISP Framework can be found in D7.3.

The main aim of recalling the C3ISP Framework reference architecture here is that in the following, we describe each of the tools that are being provided by the C3ISP partners, and for each of them this document specifies which of the components shown in Figure 1 can benefit from the tool.

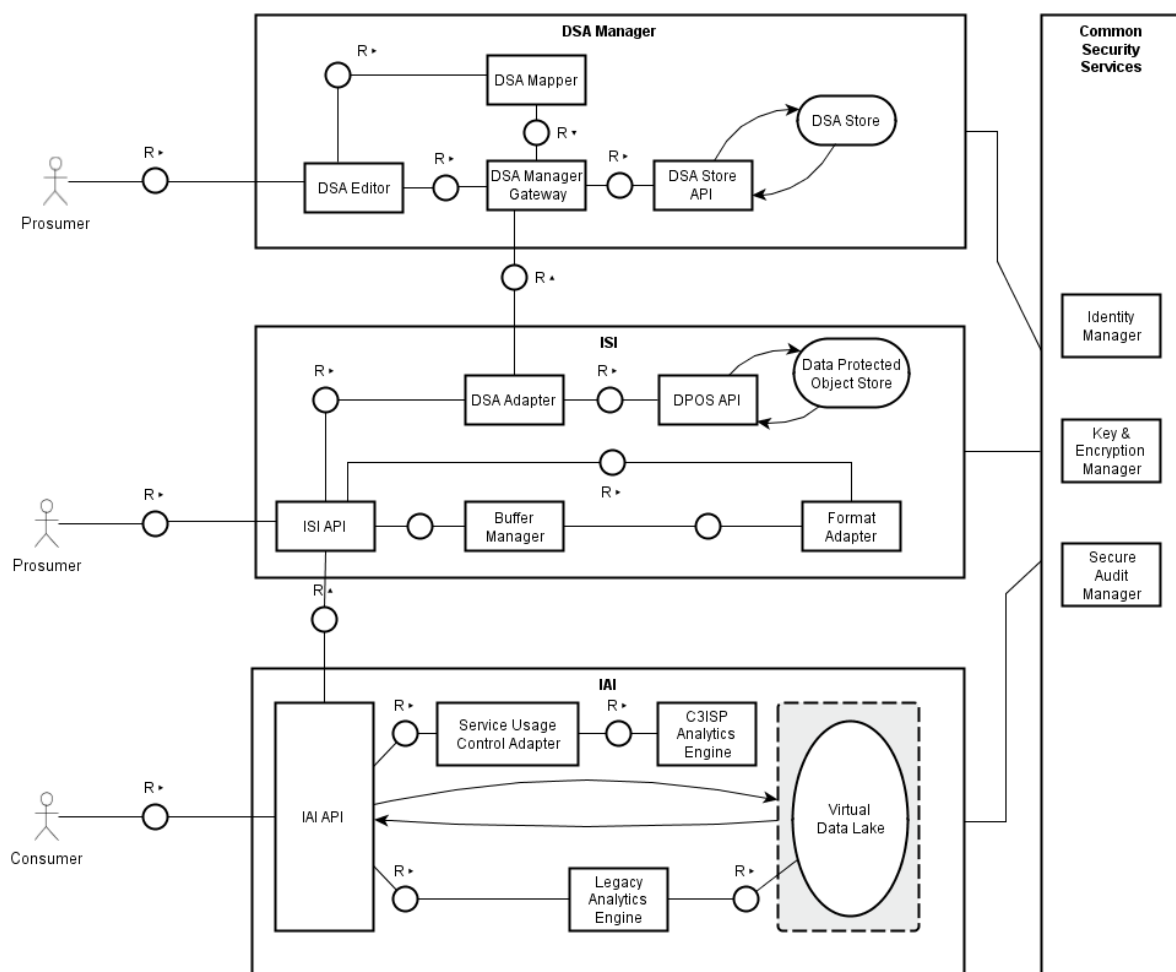


Figure 1: C3ISP high-level architecture – version Month 24, taken from D7.3 in Section 2

4. Data Sharing Agreements

A Data Sharing Agreement (DSA) is a contract regulating the sharing of data among entities (organizations, individuals, etc.). In the C3ISP scenario, DSAs are defined by the CTI producers to regulate the usage of the information in the CTIs when they shared in the data analytics processes. These DSAs concern both the CTIs shared by the CTI Producers, which are used as input of the analytics processes, and the new CTIs resulting from the analytics computation. In particular, DSAs express constraints on the shared information in terms of Data Manipulation Operations (DMO), i.e., the operations required to pre-process the shared information before its usage (e.g. to anonymize the input data before processing them or the analysis results before sharing them with the other Prosumers, and Analytics operations in order to permit or forbid the execution of an analytic service of the shared data depending on certain conditions.

Summarizing, DSAs define whether the shared CTI can be exploited to compute a given analytic operation, which manipulation operations must be performed on the related data before performing the elaboration of the analytics engine, and which manipulation operations must be performed on the results before they can be shared with the other Prosumers.

4.1. DSA Editor

4.1.1. Component description

The DSA Editor is the component used for authoring Data Sharing Agreements (DSAs), i.e. the “contracts” that specify the agreed set of policies (also called rules) used for regulating the CTI data sharing and analysis functionalities provided by the C3ISP Framework. The DSA Editor provides an interactive approach that takes advantage of the ontology [2] technology to guide the user in the definition of the DSA policies. When the user is creating a policy (see Figure 2), 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. This effectively helps the user write sound rules, like in the following example where the user is writing “IF a Subject hasRole” and the pop-up helps completion of the sentence with the correct terms:

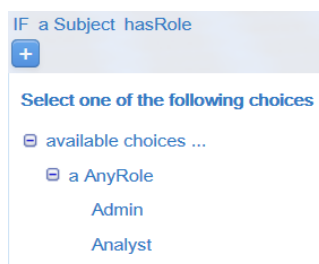


Figure 2: Steps to write a policy are interactively guided during the editing process. Only the relevant terms and actions are shown to the user helping him/her in defining sound rules. In this case, thanks to the ontology, the property “hasRole” accepts the terms “Admin” or “Analyst” only.

The vocabulary used by the DSA Editor is an ontology written in the Ontology Web Language (OWL¹). The vocabulary describes the usage context in which the DSA will be

¹<https://www.w3.org/OWL/>

used; for example, sharing CTI data might use different terms, actions and properties than those needed when sharing health data. For this reason, the DSA Editor supports using different vocabularies, i.e. different actions and terms and how they are related to each other. Figure 3 shows part of the ontology showing the terms used in the example reported in Figure 2.

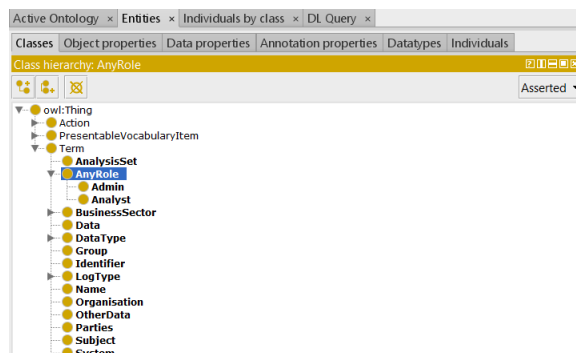


Figure 3: Part of an OWL ontology corresponding to what is shown in Figure 2, as edited in the open source Protégé²ontology editor.

As seen in the previous example, to define the policies we use a high-level language very close to the natural language. This language is based on the formal language called *CNL* – *Controlled Natural Language*[1] and allows writing easily comprehensible (by humans) 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**, **obligations** and **prohibitions** policies. These are specific types of rules to express statements respectively about what an entity **can/cannot** do, **must** do, and **must not** do.

The ontology to be used as vocabulary in the DSA Editor must import a parent ontology called *upper_vocabulary.owl* (included in the Appendix 1), since the policies are processed by a built-in parser that needs a predefined structure. The “upper vocabulary” builds this basic structure of the policies, which are made by two classes defining the main entities of the policies domain: *Action* and *Term*. Additional terms and actions must be defined as subclasses of these existing classes in any vocabulary that will be created.

In the DSA Editor, the *Term* (and its subclasses) is used as subject or object in the policies and the *Action* (and its subclasses) as verb, according to the following syntax (declared in the *upper_vocabulary.owl* through two properties):

a Term CAN/CANNOT/MUST Action a Term

For example:

a Subject CAN/CANNOT/MUST Read a Data

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

Additional properties can be defined in the vocabulary in order 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:

²<https://protege.stanford.edu/>

IF a Subject hasRole Prosumer THEN a System CAN Read a Data

Where *Prosumer* and *System* are subclasses of *Term* and *hasRole* is a property with Domain *Subject* and Range *Prosumer*.

Detailed screenshots of the tools are reported in Section 4.1.5 (First release of the component).

The DSA Editor is part of the DSA Manager subsystem and has the role of allowing users to define the rules for CTI data sharing and analysis that will be interpreted and enforced by other components running in the C3ISP Framework (in particular the DSA Adapter, see D7.2).

At the high-level, the DSA Editor defines a two-step editing workflow process. First a DSA called **Template** is created (step 1). 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. 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 the following state diagram summarises:

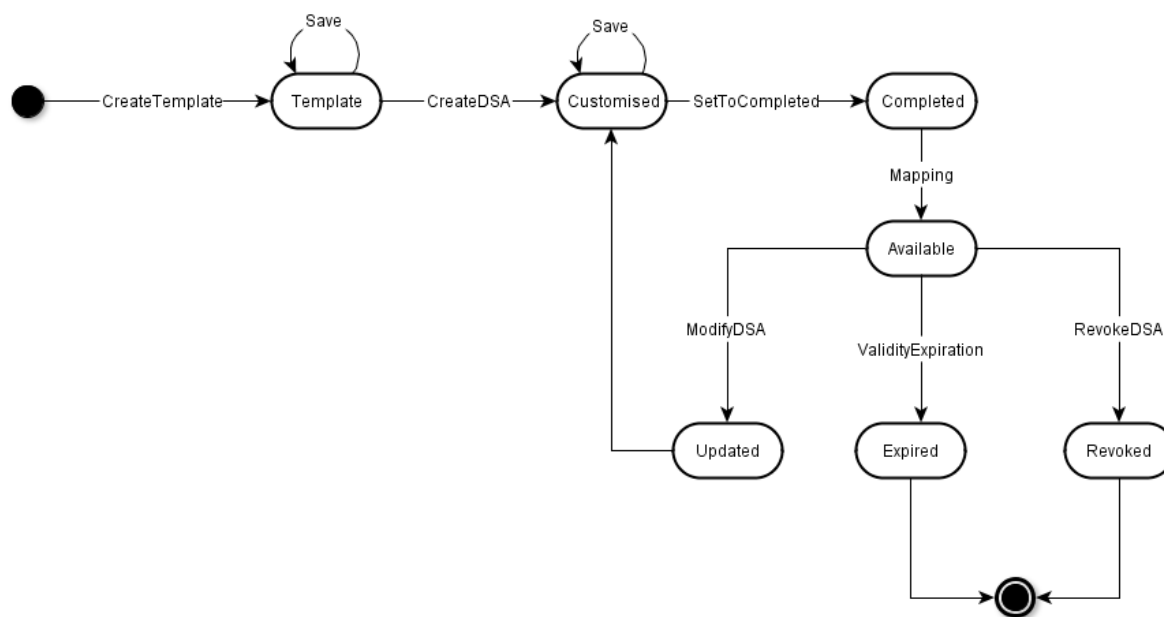


Figure 4: DSA State Diagram, refined with respect to what was reported in D8.1.

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, the user can periodically save their work, before moving to other states.

Once the editing is completed, the user selects to move the DSA (instance) to the “**Completed**” state: it means that the DSA has been finalised and can be translated to its enforceable representation (see DSA Mapper in section 4.2). When the DSA is mapped, it is

marked as “**Available**”, ready to be used for protecting data (i.e. to be paired with the CTI data).

During its lifecycle, a DSA can be “**Revoked**” (by the user or some other entity) or can be “**Expired**”, according to a validity period (i.e. a starting date/end date) specified when created, beyond which it is no more applicable to the data. An “Available” DSA might also be “**Updated**”, even if it has been already been used (i.e. paired with data). To properly manage these cases, the user can specify policies to express what happens when the DSA is revoked, expires or is being updated (see Figure 5). The behaviour for the three cases can be selected by specifying a “Deny all” policy (data access is forbidden), “Deny all and Delete now” (data access is forbidden and data will be deleted upon access), and “Delete in a specified period” (data access is forbidden, but data will be deleted after the specified number of days).

The screenshot shows three columns of policy selection: Expiration Policy, Update Policy, and Revocation Policy. Each column has a dropdown menu with three options: 'Deny all', 'Deny all and Delete now', and 'Delete in a specified period'. Below each dropdown is a 'Period in days' input field. For Expiration and Update, the value is 0. For Revocation, the value is 5.

Figure 5: The list of policies the user can select from to manage the “Expired”, “Updated”, and “Revoked” states.

In addition to the rules, the DSA also keeps **metadata information** to better define its context: it specifies the purpose of the data sharing (**Purpose**), the type of data involved (**Data Classification**), the DSA validity time interval (**Validity**) and the parties participating to the agreement (**Parties**), see Figure 6. Since the DSA Template is more generic than a DSA instance, some of these metadata might be specified at DSA Template creation time for certain use cases, while others might be defined during the DSA creation phase.

The screenshot displays the DSA metadata configuration form. Fields include:

- Title***: DSA title
- Status**: CUSTOMISED
- Purpose***: Data Breaches Notification
- Data Classification**: Highly Confidential
- Validity***: start date: 2018 Mar 2, end date: 2018 Sep 7
- Parties**: A table with columns 'Name' and 'Choose party name'. It lists '3D Repo' and 'CHINO' with dropdown menus and 'remove party' buttons. An 'add party' button is at the bottom.

Figure 6: DSA metadata fields.

The DSA Editor allows using a Term across policies; it means that the same Term can be used in several policies indicating a reference (Figure 7) to it while defining policy:

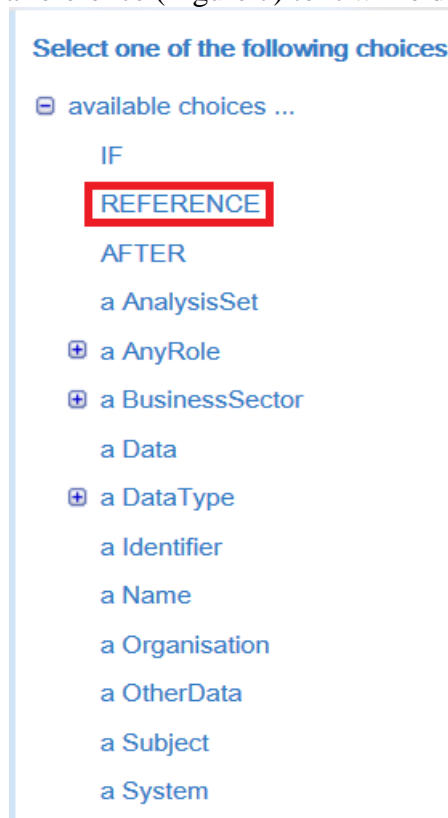


Figure 7: To create a reference between Terms, a user has to select REFERENCE in the pop-up guiding window.

A specific option (REFERENCE) in the provided pop-up is available for this purpose that, when selected, hints the available terms (highlighted in green, see Figure 8) that can be selected for completing the policy:



Figure 8: In this case the user is writing “IF a Subject hasId” and wants to use the same Term as in the circled policy (“a Identifier”). So the user after clicking on REFERENCE, can select the Term highlighted in green (“a Identifier”) to create the reference.

The result of selecting a reference in the user interface is to add “that” in the policy before the referenced Term, as shown in Figure 9:



Figure 9: References in the user interface; “that Identifier” points to “a Identifier” in the policy at the top. The DSA Editor offers also a feature to graphically show all the defined references.

Policies defined using the DSA Editor are saved in the DSA Store (as an XML file).

4.1.2. Maturation status

Previous deliverable D8.1 stated we plan to reach TRL (Technology Readiness Level) 6 by working on the following improvements:

- Define and support new vocabularies for the C3ISPPilots’ context, with particular attention to the legal aspects related to the security of the information sharing;
- Support definition of Data Manipulation Operations (DMOs) (pre/post-processing rules) [it copies the requirements of C3ISP-Fun-DS-011 and C3ISP-Fun-DA-002 described in D7.1];
- Support for Data Analytics Operations [it meets C3ISP-Fun-DA-001 requirement];
- Support notifications for results [it meets C3ISP-Fun-DS-009 requirement];
- Improvements on the DSA Editor usability [it meets C3ISP-Usa-002 requirement].

In the following sections, we describe the reached maturation status for those improvements as well as other improvements that were deemed as necessary during the project development.

4.1.2.1. *Define and support new vocabularies for the C3ISP pilots’ context*

We have defined a new vocabulary by interacting with Pilots owners and studying Pilots requirements. This vocabulary uses an ontology in which actions, terms and properties of the terms support the definition of policies in a cross-pilots domain. In fact, we tried, and as of now succeeded, to standardise the terminology for creating a single vocabulary suitable for all the Pilots’ use cases. Having a single vocabulary helps operational maintenance, allows the user a more streamlined user experience and results in a lower learning curve to get acquainted with it. It could also enable cross-pilots use case scenarios. Nevertheless, since the DSA Editor supports using any number of vocabularies (it allows selecting it when creating the DSA Template), if necessary, we could define different vocabularies for each Pilot.

The vocabulary (see Figure 10) is currently available in the C3ISP testing environment (https://dsamgrc3isp.iit.cnr.it:8443/vocabularies/c3isp_vocabulary_3.2.owl#) and Pilots are using it for creating the set of rules they need. Continuous feedbacks are reported by the Pilots’ owners and we expect to release improved versions of the ontology as the C3ISP project matures. Please see the Pilots specific deliverables for a sample of their defined DSA policies.

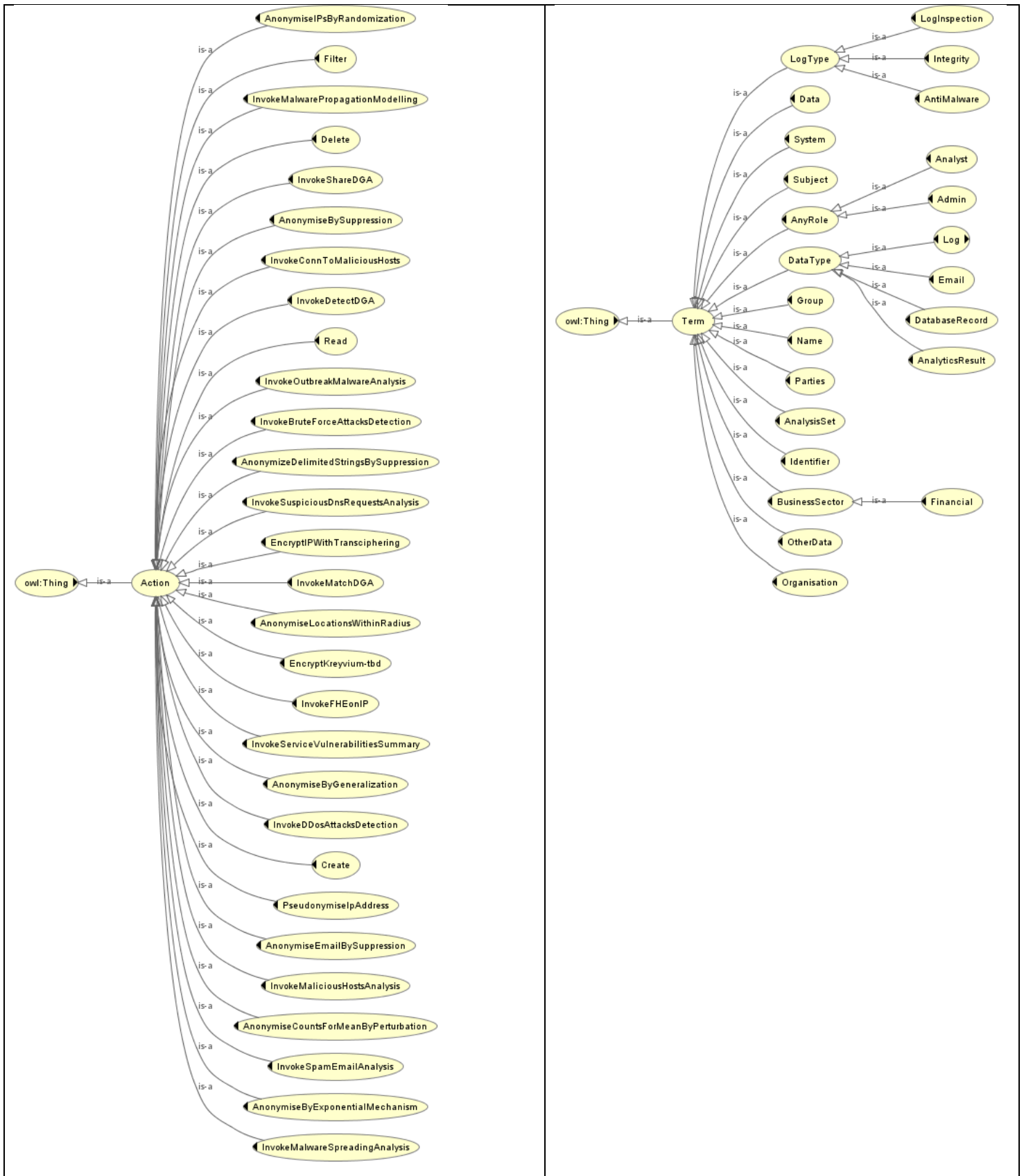


Figure 10: Graphical representation of Actions and Terms of the current vocabulary.

4.1.2.2. Support definition of Data Manipulation Operations (DMOs) (pre/post-processing rules)

Support for DMOs, in particular those defined in the SAP Anonymization Tool (see Section 8.1) has been introduced both at vocabulary-level, and at DSA Editor-level.

A DMO is specified by a name (e.g. AnonymizeDelimitedStringsBySuppression) and by a set of properties that define how it will be applied to the data. In particular, a DMO can have the following properties:

- **Parameters:** they specify the (part of) CTI data the DMO will be applied on. It contains a list of data fields (in the CEF³ format);
- **Options:** they describe how the DMO will be applied, for example by gauging the magnitude of anonymization (e.g. how many digits/octets of an IP address will be suppressed). Basically, condensed and simplified in a single and easy to understand keyword (e.g. HIGH_PRIVACY, LOW_PRIVACY, etc.), to describe the configuration settings of each specific DMO. This has been done to simplify user experience.

At vocabulary-level, a DMO is defined as an **Action** in the ontology (e.g. AnonymizeDelimitedStringsBySuppression), while options and parameters are specified as ontology **Annotations** for that Action (respectively in the *seeAlso* and *isDefinedBy* annotations). To define a new DMO or to change the parameters / options of an existing DMO, it is just a matter of acting on the vocabulary. The annotations will be interpreted by the DSA Editor to present a specific user interface and by the DSA Mapper when it will create the enforceable representation of the policy (see 4.2).



Figure 11: DMO definition in the ontology.

³CEF (Common Event Format) is the normalized format of the CTI data stored into C3ISP, see D7.2 for more.

At DSA Editor-level, a DMO is defined in an obligation policy. When the user selects a DMO from the vocabulary (e.g. AnonymizeDelimitedStringsBySuppression), s/he has the capability of adding Parameters and Options by clicking on the “Add Info” button and selecting the values as defined in the ontology annotations. Figure 12 shows this user experience.

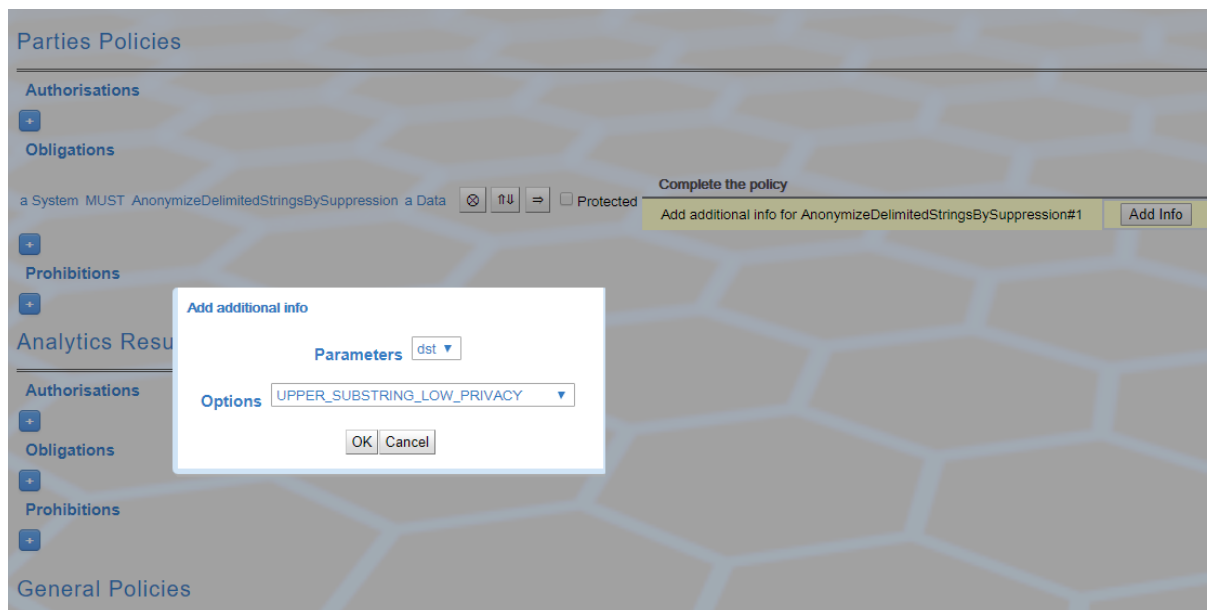


Figure 12: Fill in additional information for the DMO

4.1.2.3. Support for Data Analytics Operations

Analytics are identified by an “AnalyticsName” (e.g. ConnToMaliciousHosts) and included as Actions in the ontology. For improving the user experience in the policies definition, we adopted a naming convention: all analytics services start with the prefix “Invoke” (i.e. **InvokeConnToMaliciousHosts**). Then an analytics function can be invoked by a *Subject* on a *Data* or on a certain set of data (named “AnalysisSet”; a *Data* belongs to an *AnalysisSet*) as shown in the picture below.

Type	Policies
AUTHORIZATION	IF a Subject hasRole Admin THEN that Subject CAN InvokeShareDGA a Data

Figure 13: Policies for handling rules on Analytics services

Analytics parameters are configuration settings to be used at analytics execution time. To simplify the end-user experience, those parameters are not defined in the policies (or vocabulary), but in specific analytics service configurations (hosted by the C3ISP Analytics Engine). For this reason, we opted to have **different analytics names defined in the ontology to accommodate different analytics configurations** (e.g., InvokeDetectDGA, InvokeShareDGA, InvokeMalwarePropagationModelling, InvokeMalwareSpreadingAnalysis, etc.). Since the number of analytics is not big and the same applies to the number of configurations for each analytic, we think this is a good trade-off that helps the user in writing more easily the DSA policies.

4.1.2.4. *Support notifications for results*

Support for the notifications for results have not been added yet. It will be part of the activities for the next period.

4.1.2.5. *Improvements on the DSA Editor usability*

At the DSA Editor-level, some changes have been introduced in order to create a more relevant DSA according to the pilots' domains and use cases.

In particular:

- Fields not relevant for the C3ISP Pilots have been removed from the original DSA Editor prototype, polishing the user interface;
- roles and responsibilities of the parties;
- indemnities;
- governing law;
- data subject policies;
- A new list of values for the Purpose (law obligations, cyber monitoring, cyber investigation, contractual obligations) and Data classification (confidential, highly confidential, public, operational data) has been defined. These values have been moved to a configuration file for easier maintenance;
- When creating a new DSA Template, the choice of the vocabulary is delegated to the user that can select from a drop-down menu (before it was a free input text field asking for a URI). The list of vocabularies has been defined in a configuration file;
- Improvements to the stylesheets used to show the DSA Editor (C3ISP background image and logo, colours, etc.);
- Finally, the DSA Editor supports internationalization (I18N): the tool is already available in English (the default), Spanish, Italian and French, and additional languages can be added with little effort. Updated DSA Editor's functionalities have extended the messages and labels used for translations (translations are not hardcoded but loaded dynamically).

4.1.2.6. *Other improvements*

4.1.2.6.1. New section for Policies on analytics results

We added a new section to the DSA Editor screen, called “Analytics Result Policies”, to allow the user 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 used for the shared data: Authorization, Obligation, Prohibition (see next figure).

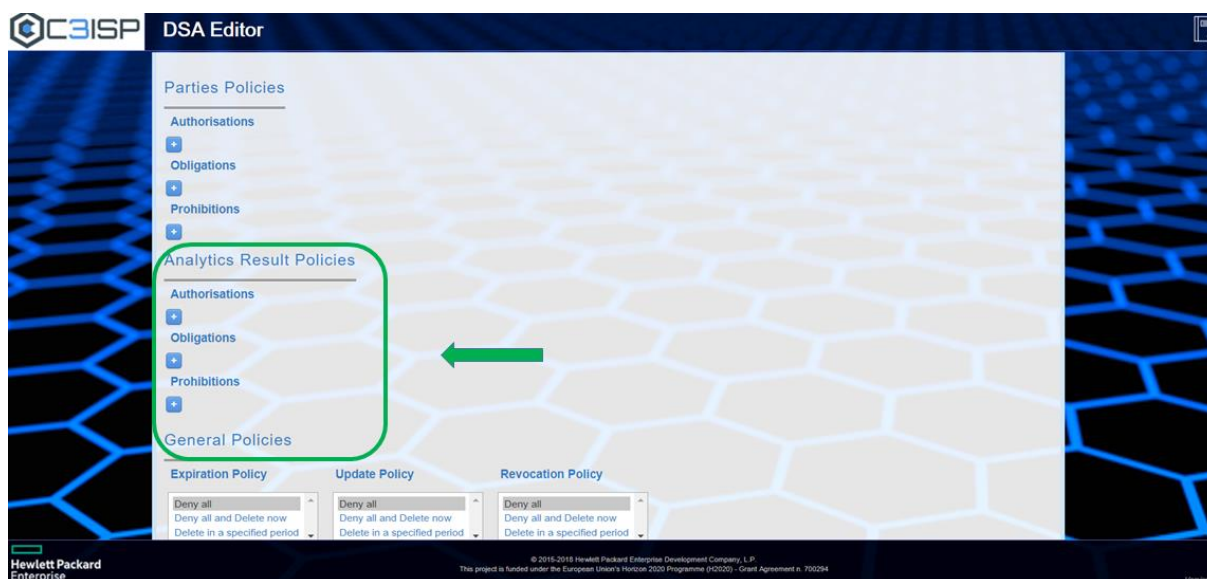


Figure 14: New section: Policies on Analytics Result.

This is a first version and we plan to improve it in the next period. Currently, this choice for the policies on the analytics result has the following assumptions and raises some questions to be investigated better later:

- We use the same policies for all the derived data, i.e. all the analytics result;
- When there are no policies for the results specified in the DSA, the default is to permit access to result data.

4.1.2.6.2. Use of free text fields

The DSA Editor supports specifying free text fields to be inserted by the user. This is used for defining policies where there is a free text field like an identifier. In fact, this feature is applied to the “Identifier” Term in the vocabulary, where we set a specific **Annotation** (*isDefinedBy* set to User).

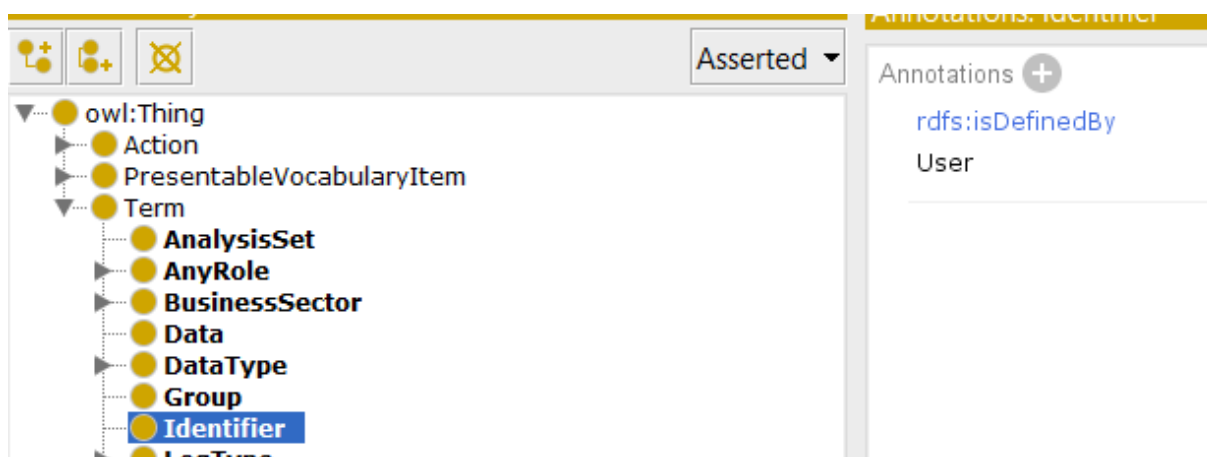


Figure 15: Annotation used to handle free text fields in the DSA Editor for the Term “Identifier”.

When the Term with this annotation is used, the DSA Editor displays a free text field to be filled in:



Figure 16: Fill in additional information for a value defined by the user on the “Identifier” Term.

4.1.2.6.3. New Property for evaluating membership of a Subject to a Group

During the past period, we received a specific Pilot requirement to support writing DSA policies that can check whether or not a user (subject) is member of a group of users. Leveraging on the free text field support described above, we added the Term “Group” to the vocabulary with a Property called “isMemberOf”, and set the *isDefinedBy* annotation to User:

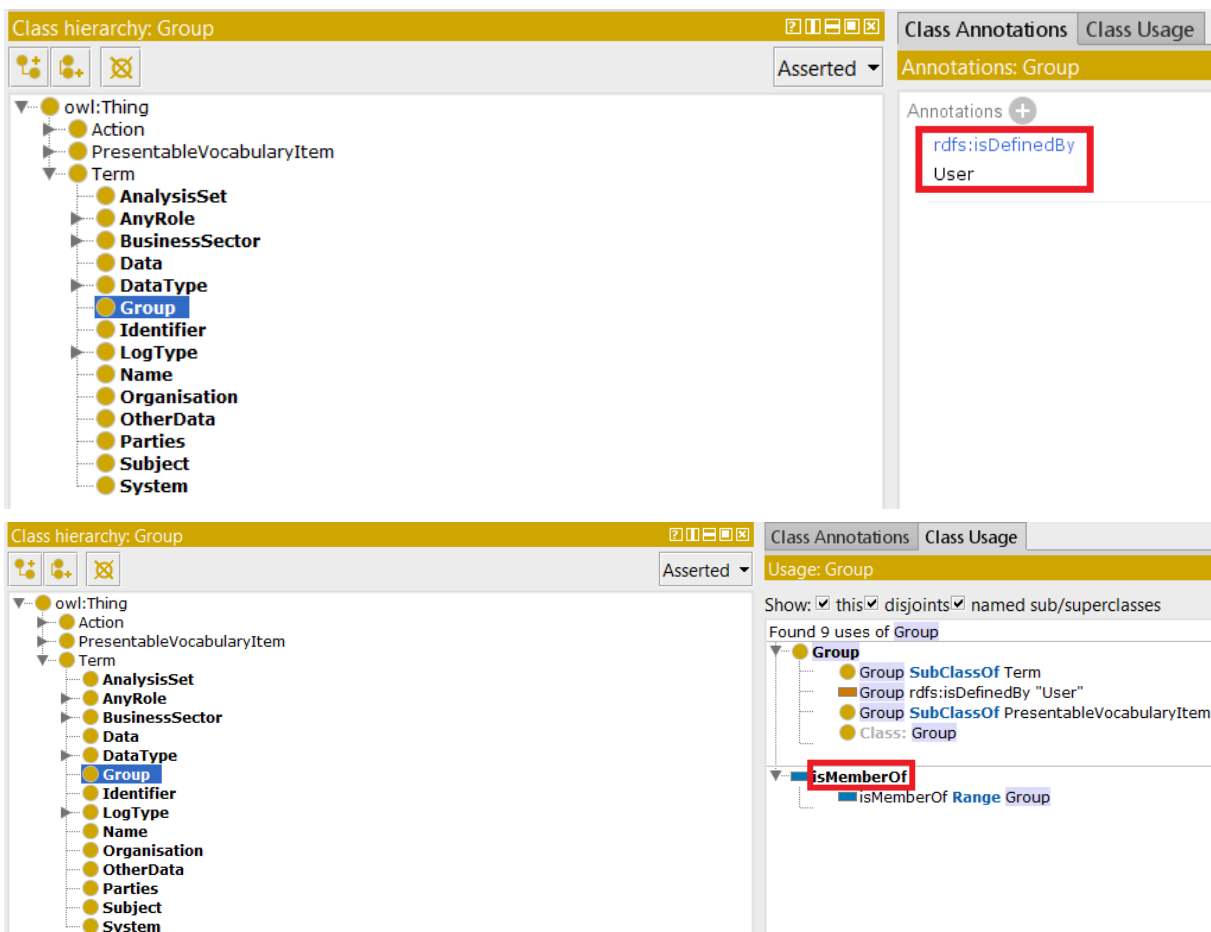


Figure 17: “Group” and its annotation definition in the ontology (top image). The “Group” Term has a “isMemberOf” property associated (bottom image).

This allows writing a sentence to evaluate if a “Subject” is a member of a specified “Group”. It needs as input parameter the Group name (the free text manual user entry), as in the following examples:

```
AUTHORIZATION    IF a Subject isMemberOf a Group(Group One) THEN that Subject CAN Read a Data
AUTHORIZATION    IF a Subject isMemberOf a Group(Group Two) THEN a Subject CAN Create a Data
```

Figure 18: “isMemberOf” in a DSAauthorization policy; Group One and Group Two are the group names.

“isMemberOf” works in a similar way to “hasId” (used in the free text field case) that evaluates a Subject based on his “Identifier”.

At enforcement time, group membership will be evaluated by using the specified group name and the subject identity (e.g. by checking the Subject’s LDAP groups).

4.1.2.6.4. Updates to the Role-based Access Control settings

In order to support the DSA lifecycle (see DSA state diagram in Section 4.1), the access and the use of the DSA Editor follows a role-based access control model (RBAC [3]) 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 finalise the DSA with the specific business rules.

The following table shows the permissions based on the role. 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, see 4.2), and Revoke (to set the DSA into the “Revoked” state).

Role	Action	DSA states						
		Template	Customised	Completed	Available	Updated	Expired	Revoked
Legal expert	Create	X						
	Read	X			X			
	Update	X						
	Delete	X						
	Complete							
	Revoke							
	Map							
Policy expert	Create		X					
	Read	X	X	X	X	X	X	X
	Update		X		X	X	X	
	Delete		X	X	X	X	X	X
	Complete		X			X		
	Revoke		X	X	X	X	X	

	Map			X				
--	-----	--	--	---	--	--	--	--

4.1.3. Addressing specific privacy requirements in DSAs

C3ISP DSAs allow codifying policies that can fulfill a broad range of privacy requirements. In particular, their expressive power includes:

- Granting or denying access to shared CTI data, which could include for example personal data;
- Controlling access and usage of analytics functions and the analytics’ result (result could include privacy-related data, as well);
- Defining Data Manipulation Operations (DMOs) on the shared data, to process the data to assure the desired level of privacy.

By combining these features, with the help of the Ethical Advisor and other experts in the field, it would be possible to encode the privacy requirements expressed by the Pilots and beyond. In the following sections, we expand on the notions cited above.

4.1.3.1. Granting or denying access to shared CTI data

A DSA can define policies that take into account access control restrictions. Based on those restrictions, it is possible to grant or deny access to the shared CTI data to only users matching them. For example, they can easily express access criteria based on user’s profile (e.g. a user must have a specific profile attribute value to access the shared data), group membership (e.g. a user must be in a specific group of users or in more groups), role-based access control – RBAC (e.g. a user must have a certain role(s) for access).

C3ISP also extends the classic access control paradigm allowing usage control, i.e. it can continuously check the access rights validity after access has been initially granted (see Section on Continuous Authorization Engine);

4.1.3.2. Controlling access and usage of analytics and the analytics’ result

We can define policies that control the access and use of the analytics services with the same criteria we can use for access control for protecting the shared data. But in addition to that, we can define policies that put privacy restrictions about what to do with and how to share the result data object generated by the running of an analytics function. The result data can have its own policies or inherit the “parent data policies” (i.e. the policies used by the data processed by the analytics) and this depends on how we defined the DSA policies (see Section 4.1.2.4).

4.1.3.3. Defining DMOs on the shared data

This includes anonymization functions that aim at protecting individual privacy, while at the same time maintaining enough value and significance in the data for the analytics algorithms. Further, C3ISP DMOs supports also different forms of encryption, both well-known standard-based and advanced homomorphic-based. Section 4.1.2.2 describes the currently available DMOs.

4.1.4. Requirement Analysis at M24

Table 1 – DSA Editor Requirements Status

ID	MET	Description
C3ISP-Com-DE-001	YES	The tool supports the definition of Data Sharing Agreements between parties. For defining policies on CTI data and analytics results we defined a vocabulary (based on an ontology) for expressing these constructs and operations on them.
C3ISP-Com-DE-002	YES	The tool supports the definition of multi-lateral Data Sharing Agreements that is two or more parties can be specified in the agreement.
C3ISP-Com-DE-003	PARTIALLY	The definition of a specific vocabulary is needed for expressing policies and legal constraints on CTI and analytics results. First version of vocabulary is defined, and improved version will be released based on Pilot's requirements. DSA Editor has been enhanced to support specific policies on analytics results.
C3ISP-Com-DE-004	YES	It allows defining access and usage control policies.
C3ISP-Com-DE-005	YES	It allows defining policy that requires a parametric value for some fields.
C3ISP-Com-DE-006	PARTIALLY	The definition of a specific vocabulary is needed for expressing data manipulation operation. Vocabulary includes DMOs as actions. Based on Pilot's requirements, it could be added/refined version of vocabulary actions.
C3ISP-Com-DE-007	YES	The tool already uses Web ontologies for defining its vocabularies.
C3ISP-Com-DE-008	YES	It already allows for a DSA validity period.
C3ISP-Com-DE-009	YES	The tool allows editing DSA and now also supports revoking DSA.

CEISP-Comp-DS-010	YES	The tool is a Web application available also <i>as a service</i> .
C3ISP-Com-DE-011	YES	The tool allows interactive and guided policy authoring.

4.1.5. First release of the component

The DSA Editor component is a web tool reachable from the Internet at the following link:

<https://dsamgrc3isp.iit.cnr.it/DSAEditor/>

It is hosted on a Virtual Machine at CNR’s Farm, where we setup the C3ISP testing environment. The tool is written in Java, it uses Vaadin⁴ and GWT⁵ technologies for the web user interface presentation, and the Spring Framework⁶ for the backend logic. Deliverable D7.3 contains a description of the DSA Editor internal architecture.

Once connected to the DSA Editor, it asks for user authentication:

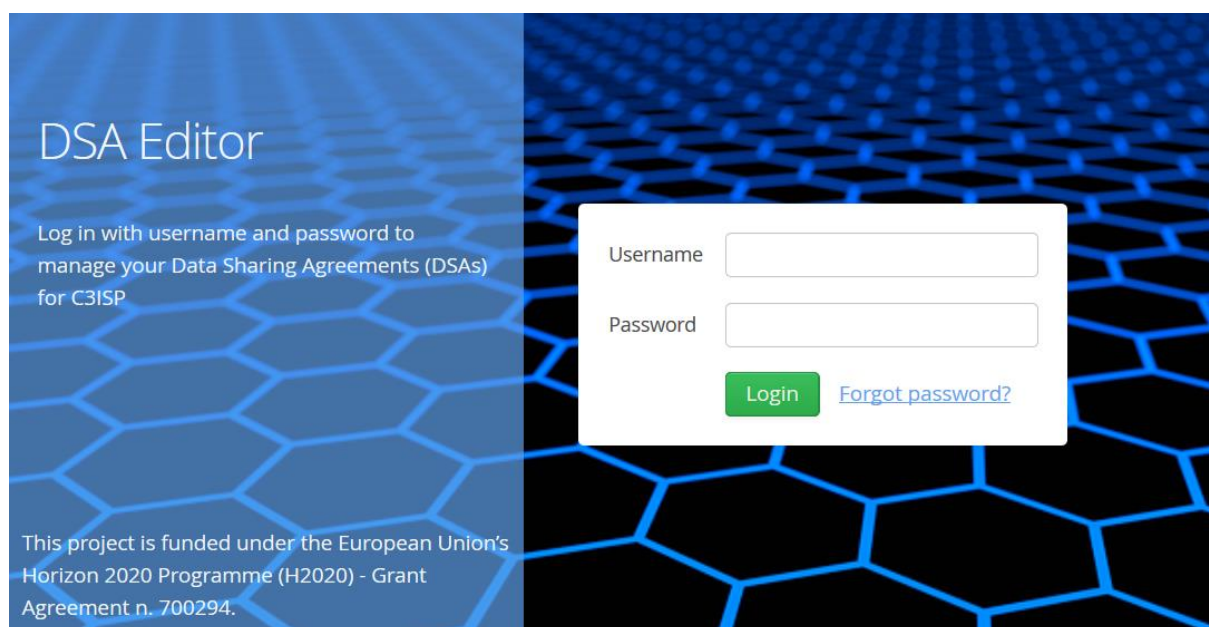




Figure 19: DSA Editor Tool: Login Screen.

As reported in 4.1.2.6.4, users’ permissions are based on roles. Each user is assigned either the “lawexpert” role (used to create the DSA Template; the  button is shown on the top-right corner), or the “policyexpert” role (used to create the DSA instance; the  button is shown on the top-right corner). DSA Editor’s users along with their roles are stored in an internal database based on MySQL⁷.

⁴Website: <https://vaadin.com>, open source code: <https://github.com/vaadin/platform>

⁵Google Web Toolkit (GWT): <http://www.gwtproject.org>

⁶Spring Framework: <https://spring.io>

⁷MySQL Community Server: <https://dev.mysql.com/downloads/mysql>

Once connected, the user sees the list of his/her DSAs and, depending on the role, specific functionalities are available or not (see RBAC matrix in Section 4.1.2.6.4):

DSA Name	DSA Creator	DSA Version	DSA Status	Create Date	Start Date	Expire Date	DSA ID
Anonymize new	policyexpert	1	Available	2018-07-26	2018-07-26	2020-03-31	DSA-73556061-9021-49fb-ac9d-4fda4638934f
DSA with OtherData Policies	policyexpert	0	DSA Is Expired	2018-07-20	2018-03-02	2018-03-31	DSA-0f7b1aab-bd2a-4380-a629-602c5124cf00
DSA with OtherData Policies	policyexpert	0	DSA Is Expired	2018-07-20	2018-03-02	2018-03-31	DSA-391cadab-e88b-40eb-9bb7-0c663f650c10
DSA with Analytics policies	policyexpert	1	DSA Is Expired	2018-07-20	2018-03-02	2018-06-23	DSA-f8e4cd5d-257d-49f0-ae40-cef821fc8f21
DSA with OtherData Policies	policyexpert	0	DSA Is Expired	2018-07-20	2018-03-02	2018-03-31	DSA-fae08104-09c8-4981-a342-c139e484338
new untitled DSA	policyexpert	0	DSA Is Expired	2018-07-20	2018-07-20	2018-07-20	DSA-8376cddb-6ba7-4212-89d8-140846644a4d
PAOLO - Test 1 - policy other data	policyexpert	1	Available	2018-07-20	2018-07-20	2027-12-25	DSA-b74fee5a-8ab1-4ab1-b4f0-6ea776882662
Paolo - Test 2 - Subject organization	policyexpert	1	Available	2018-07-20	2018-07-20	2027-12-25	DSA-c3602f5-98f3-40ab-b475-20af0ee2b1f8
DSA with DMO policies	policyexpert	0	Customised	2018-07-11	2018-03-02	2018-09-07	DSA-7a4a3c73-fa0e-49ce-859e-ca07815cb8eb
DSA with Analytics policies	policyexpert	0	DSA Is Expired	2018-07-11	2018-03-02	2018-06-23	DSA-ba82b658-2557-42a1-979a-94ce08a0186b
DSA with OtherData Policies	policyexpert	0	DSA Is Expired	2018-07-11	2018-03-02	2018-03-31	DSA-deeec8ac-a4ae-4f9f-9ec7-3b84d4dc3695
Rome meeting DSA	policyexpert	0	DSA Is Expired	2018-07-11	2018-03-08	2018-03-08	DSA-4da7721d-d373-4f9b-a4a9-c6579a1a39c
test bugs	policyexpert	0	DSA Is Expired	2018-07-11	2018-03-16	2018-03-16	DSA-daf8e6bd-044e-4f0b-b163-07bd2ae1ea58
my test dsa 20180327	policyexpert	0	Customised	2018-07-11	2018-03-27	2019-11-01	DSA-5ae7fa02-435b-4e8c-9195-b3bf9ac8af69
PAT Test Template 24Apr	policyexpert	0	DSA Is Expired	2018-07-11	2018-04-24	2018-07-27	DSA-2317b6b8-db47-4b5e-9f31-7e9c4aac84a2
test hasid	policyexpert	0	DSA Is Expired	2018-07-11	2018-05-11	2018-05-11	DSA-73c43ccf-31fc-4c9a-b5a6-101eb45555dd
Test isMemberOf Group	policyexpert	0	DSA Is Expired	2018-07-11	2018-05-30	2018-06-14	DSA-5cc30c6a-1a37-4649-bf0f-aa57d5c1fed7
Test Has ID 10lug	policyexpert	0	Customised	2018-07-11	2018-07-10	2020-03-23	DSA-417aa745-5850-4182-98c7-82a237797859
mirko - 20180711_1	policyexpert	14	DSA Is Expired	2018-07-11	2018-07-10	2018-07-26	DSA-c708e18c-20dd-4e77-bcad-51e3478f16c5
DSA Template isMemberOf	policyexpert	0	Customised	2018-07-11	2018-07-10	2020-03-27	DSA-dc4b9478-323e-41ba-8557-1c349898c9ee
mirko - 20180710_1	policyexpert	5	DSA Is Expired	2018-07-10	2018-03-02	2018-06-23	DSA-0fcc07c4-f3d9-4045-adfe-8a7533b7a5c9
DSA with Analytics policies	policyexpert	0	DSA Is Expired	2018-07-10	2018-03-02	2018-06-23	DSA-c50e3452-380d-4a71-bd30-5edc2d8d2455
DSA with Analytics.policies for ISP	policyexpert	3	DSA Is Expired	2018-07-10	2018-03-02	2018-06-23	DSA-e6d02745-78d9-4102-aa8e-325f12f3ea13

Figure 20: DSA Editor Tool: DSAs List.

For each DSA, the following information are reported:

- DSA Name: the name set for the DSA;
- DSA Creator: the name of the user that created it;
- DSA Version: each time a DSA is updated, its version changes (this allows managing the **Updated** DSA state and - at enforcement time - verify if the data is coupled with the latest DSA version);
- DSA Status: the state of the DSA, as per the DSA state diagram (see 4.1.1);
- Create Date: when the DSA has been created;
- Start/End Date: the validity interval of the DSA (written in the DSA itself; it allows handling the **Expired** DSA state at enforcement time);
- DSA ID: the DSA UUID⁸ internal identifier.

⁸ Universally Unique IDentifier: https://en.wikipedia.org/wiki/Universally_unique_identifier

Clicking on a specific DSA, shows the available actions in a panel on the right:

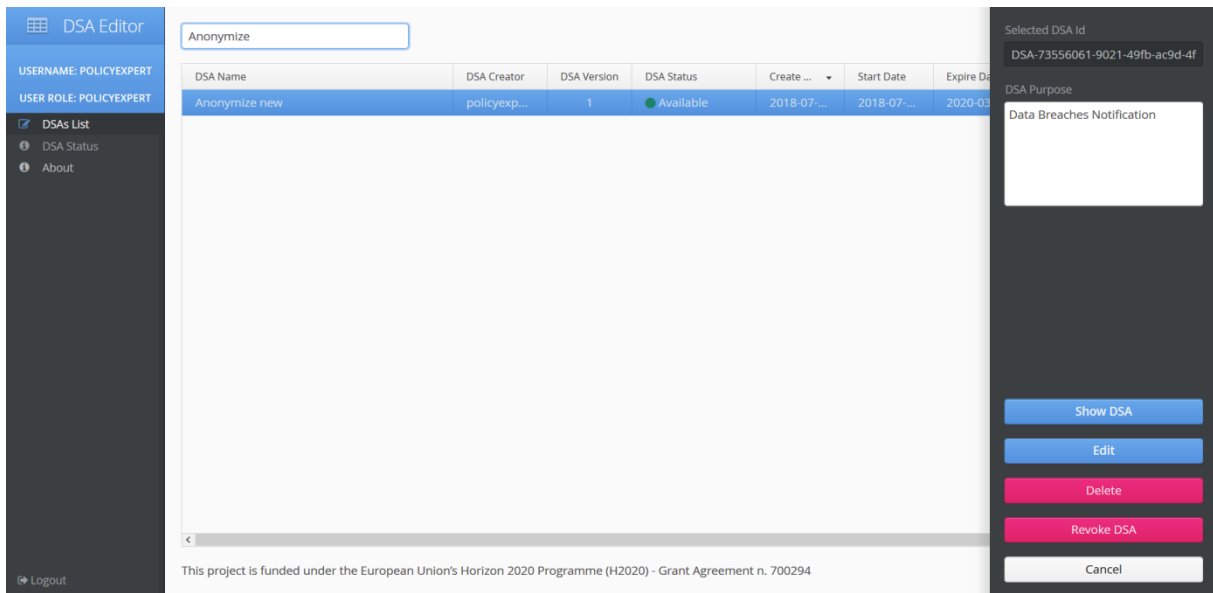


Figure 21: DSA Editor Tool: Available DSA actions.

Clicking on **Show DSA** opens the DSA Editor interface in view mode that allows exploring the DSA structure, including DSA metadata (on the top) and DSA policies (represented as CNL expressions):

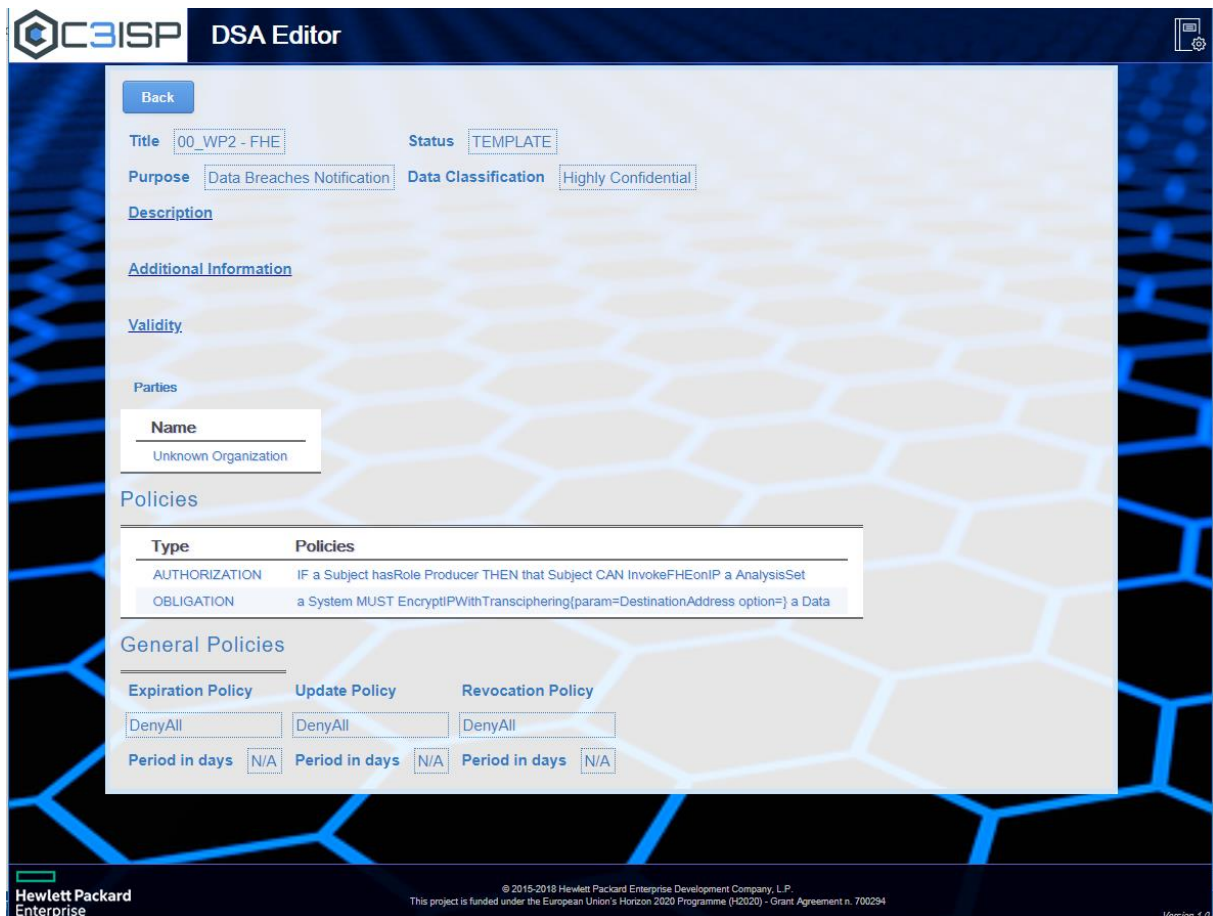


Figure 22: DSA Editor Tool: Show DSA.

Clicking on the **Delete** button removes the DSA from the DSA Store, while when selecting **Revoke**, the DSA Editor moves the DSA to the Revoked state. In both cases, a confirmation dialog is shown before proceeding.

The **Edit** button opens the DSA Editor in authoring mode and allows performing changes, like updating the DSA metadata or working on the DSA policies.



Figure 23: DSA Editor Tool: Edit DSA.

Clicking on the  button on the top-right corner, opens a dialog that allows you to select the DSA Template to instantiate (all the DSA Templates will appear, by name):

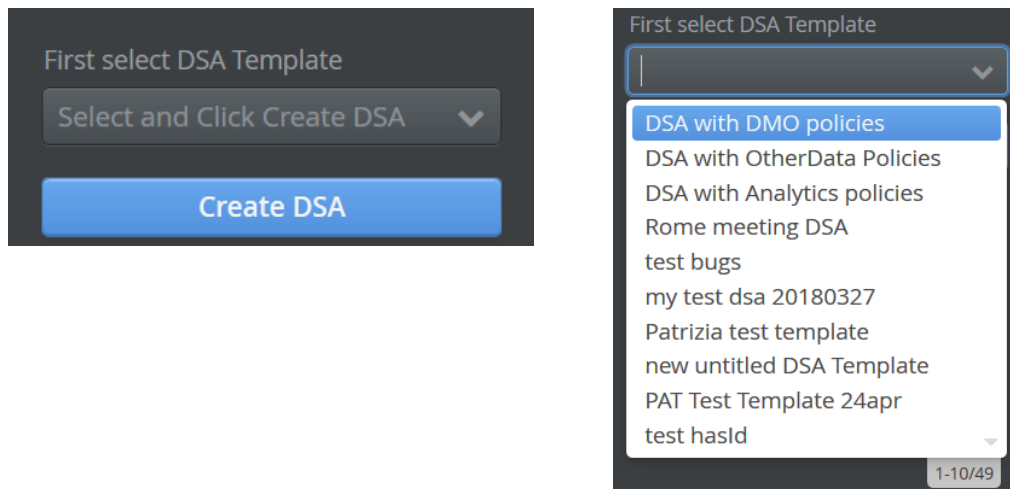


Figure 24: DSA Editor Tool: New DSA – select DSA Template; on the right a sample DSA Template list.

Once a DSA Template has been selected, the DSA Editor will show the structure of the new DSA. The new DSA will inherit the definitions already inserted in the DSA Template, including metadata and policies, if any.

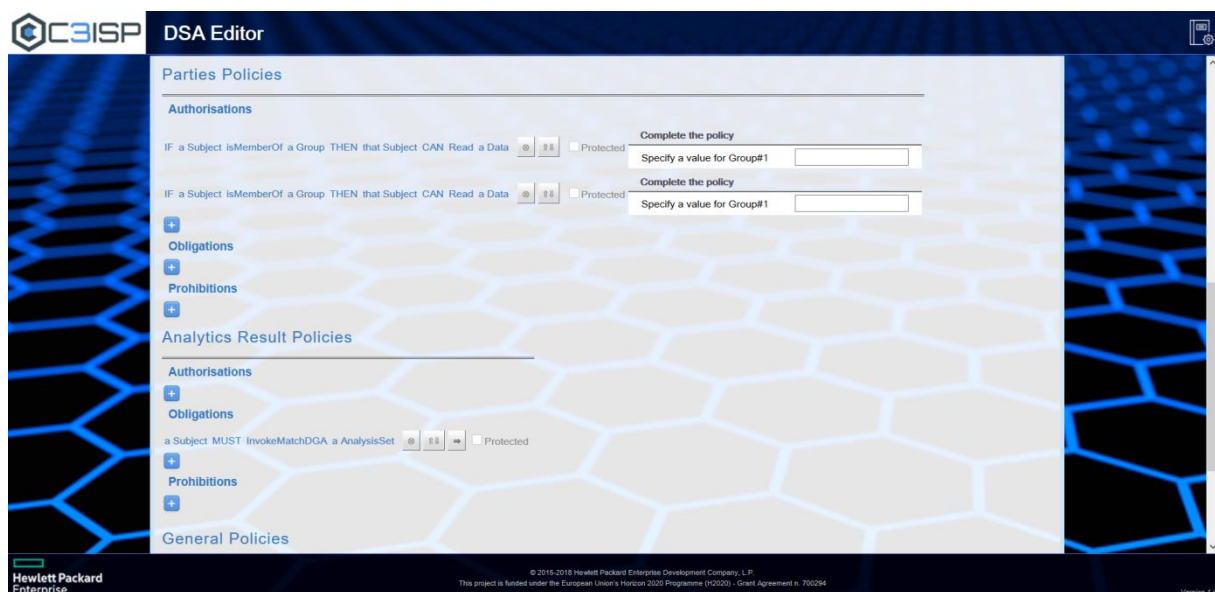



Figure 25: DSA Editor Tool: New DSA.

As explained in Section 4.1.1, the policies inherited from the DSA Template cannot be removed (the  button is greyed out), but they can be completed if there are free text fields (like in Figure 25).

With the appropriate role, a user can create a DSA Template by clicking on the **New DSA Template** button on the top-right corner. Conversely, with respect to the DSA instance, the DSA Template requires the selection of a vocabulary to load all the Actions, Terms and Properties that can be used to build the DSA policies:

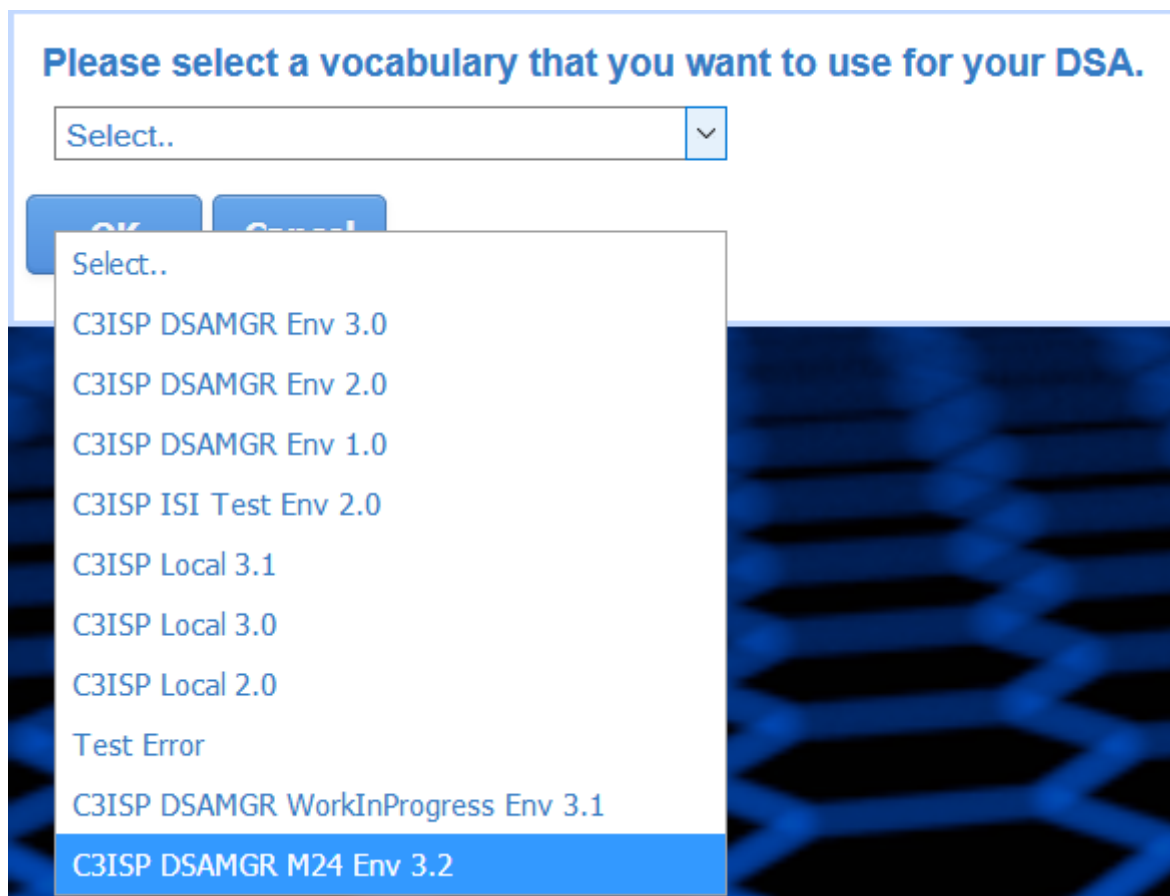


Figure 26: DSA Editor Tool: New DSA Template – vocabulary selection.

For this first release, the vocabulary defined is **C3ISP DSAMGR M24 Env 3.2** (which corresponds to what is reported in Section 4.1.2.1).

4.1.5.1. Link to Source Code

Project source code is managed using GIT SCM software and it is available at the following link:

<https://devC3ISP.iit.cnr.it:8443/c3isp-wp8/dsaManager/dsaEditor/dsaat.git>

4.1.5.1. Source Code Description

The DSA Editor is a web tool accessed via a web browser; a list of the internal APIs and their description is reported in Section 7.4.

The DSA Editor source code structure is depicted in Figure 27:

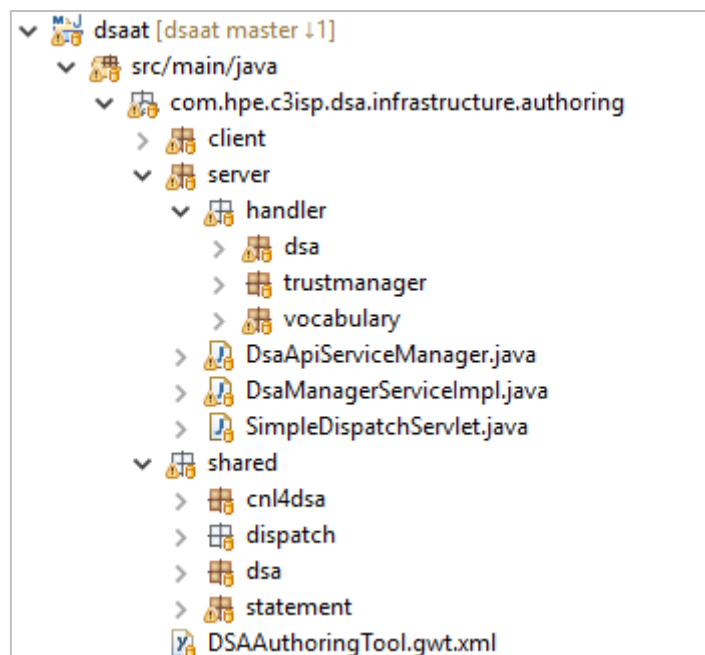


Figure 27: DSA Editor code structure

Build tasks and project dependencies (Spring, GWT and Vaadin frameworks and other utilities libraries, as described in 4.1.5) are managed using Maven.

The package *client* contains all the classes implementing the presentation logic, which is based on the GWT framework. In addition to the UI widgets and layout definitions, this package also contains the application language localisation configurations. The basic GWT application configuration is contained in the file `DSAAuthoringTool.gwt.xml`, under the root project directory.

The package *server* contains the classes implementing the backend services, including both the DSA Editor application handlers and the interaction with remote DSA API services.

Finally, the *shared* package contains the application models and main logic shared between the classes under *client* and *server* packages. The fundamental class in this package is `/shared/dsa/DsaBean.java`, which contains the model representing a DSA object including all its basic properties, metadata and policies. This model is used both for the DSA presentation and editing tasks in the UI and DSA serialization and communication tasks operated in the backend.

4.2. DSA Mapper

4.2.1. Component description

The DSA Mapper aims to transform the set of rules of a DSA, defined through the DSA Editor (Section 4.1) and specified in Controlled Natural Language (CNL), into policies that can be automatically enforced by the Continuous Authorization Engine (CAUTHENG) described in Section 5.7.

This component has been designed and developed within the Coco Cloud EU FP7 project in order to provide a mapper function that is suitable to cope with all vocabularies and DSA provided by the use cases. In the C3ISP Framework, the DSA Mapper is part of the DSA Manager component, and the DSA Mapper prototype released in Coco Cloud EU FP7 project

will be extended to cover the C3ISP requirements. The current Technology Readiness Level (TRL) of the DSA Mapper is 4, and within the C3ISP Framework we plan to mature it to reach TRL 6.

4.2.2. Maturation status

With respect to what has been described in D8.1, the DSA mapper has been improved coherently with the DSA Editor. In fact, as well as, new functionalities, kind of policies, and new vocabularies have been introduced and managed at the level of the DSA Editor, the DSA Mapper has been enhanced to manage them. Hence, the improvements are related to:

- support new vocabularies for the C3ISPPilots' context, with particular attention to the legal aspects related to the security of the information sharing [C3ISP-Com-DM-004];
- support definition of Data Manipulation Operations (DMOs) (pre/post-processing rules) [it copies the requirements of C3ISP-Com-DM-006];
- support for Data Analytics Operations [C3ISP-Com-DM-004];
- translate policies on Analytics Results.

4.2.2.1. Support new vocabularies for the C3ISPPilots' context

As reported in Section 4.1.2.1, new vocabularies have been defined for the four C3ISP project. The DSA mapper is able to learn them and recognize new terms in each vocabulary in such a way to correctly map DSA policies.

4.2.2.2. Support definition of Data Manipulation Operations (DMOs)

Data Manipulation Operations are managed at vocabulary-level as an Action of the ontology where options and parameters are managed as Annotation for the specific action. At the level of DSA Editor, DMO are obligation policies.

The DSA Mapper is able to recognize DMO policies and to traduce them as obligations policies in which the subject is always the term "System".

Let us consider that in a rule (authorisation/prohibition/obligation) a DMO is referred as

```
statementInfo="AnonymizeByDelimiter{param=DestinationAddress
option=SUBSTRING_MEDIUM}"
```

The mapper returns an ObligationExpression such as:

```
<ObligationExpressions>
    <ObligationExpression                                FulfillOn="Deny"
ObligationId="AnonymizeByDelimiter%7Bparam%3DDestinationAddress+opti
on%3DSUBSTRING_MEDIUM%7D" />
</ObligationExpressions>
```

4.2.2.3. Support for Data Analytics Operations

As DMO, also Analytics are identified by an "AnalyticsName" that is included in the vocabulary as an Action. Differently from DMO, the Analytics parameters are configuration settings that can be used at analytics execution time and are not visible in the DSA editing phase.

The DSA Mapper translates Analytics in the same way it does for the other actions.

4.2.2.4. *New section for Policies on Analytics Results*

According to the xsd schema of the DSA, authorisation, prohibition, and obligation policies related to data derived from an analytics action, are collected in a separate section. The Mapper is able to identify this section and translate the policies in it in such a way that this distinction is respected.

This functionality will be improved within the third year of the project, in order to deal with not only first order analytics results but also to second order or deeper results.

4.2.3. Requirement Analysis at M24

With respect to the requirements table in D8.1 and recalled below, the DSA Mapper has been improved by fully implementing:

[C3ISP-Com-DM-004] the capability of learning new pilots' vocabularies (from Partially to YES).

[C3ISP-Com-DM-006] the capability of translating as pre-obligations policies related to data manipulation operation from PARTIALLY to YES).

Table 2 – DSA Mapper Requirements Status

ID	MET	Description
C3ISP-Com-DM-001	YES	The current version of the Mapper already grants Prosumers that the translation of data sharing constraints is compliant and consistent from the high level to the low-level specification.
C3ISP-Com-DM-002	YES	The current version of the Mapper provides as output directly enforceable access control policies.
C3ISP-Com-DM-003	YES	The current version of the Mapper provides as output directly enforceable usage control policies.
C3ISP-Com-DM-004	YES	The Mapper needs to learn every new vocabulary in order to correctly interpret it.
C3ISP-Com-DM-005	PARTIALLY	The Mapper needs to be upgraded to be able to translate as post-obligations policies related to notification that are triggered once the analytics service result is available.

C3ISP-Com-DM-006	YES	The Mapper needs to be upgraded to be able to translate as pre-obligations policies related to data manipulation operation.
C3ISP-Com-DM-007	NO	The Mapper should be matured to interpret rules about the risk of data sharing in a proper way
C3ISP-Com-DM-008	YES	The current version of the Mapper is already able to translate policies from CNL into a XACML-based language.

4.2.4. First release of the component

The DSA Mapper component is a web service, available at <https://dsamgrc3isp.iit.cnr.it:8443/dsa-mapper/swagger-ui.html>

It implements APIs by exposing endpoints developed with the RESTful paradigm. It exposes two main functionalities as it is described in D7.3 Section 7.2. Both functionalities have been implemented by using the Eclipse environment, version oxygen2.

DSA Mapper API
API for Mapping DSA and update via DSAAPI
Created by Lunardelli Alessio - CNR IIT Pisa
See more at www.iit.cnr.it
[Contact the developer](#)

dsa-mapper-service-implementation : Dsa Mapper Service Implementation

GET /v1/mapper/MapDSABYID/{dsaid}
Fetch unmapped DSA from DSAAPI, map every rule and appending the mapped rule to appropriate element in XML. After it's complete the mapped dsa it's uploaded back to repository through DSAAPI. In the end it change the dsa status to MAPPED, again through DSAAPI update status.

POST /v1/mapper/buildUPOL/{dsaid},{dsaType}
Fetch mapped (or unmapped) DSA from DSAAPI and build Standard UPOL or protected UPOL depending by dsaType parameter. Return UPOL XML in the body response. (In case of unmapped DSA, the dsa it's mapped before proceeding).

[BASE URL: /dsa-mapper , API VERSION: 1.1]

4.2.4.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/dsaManager/dsaMapper/mapperDSA>

4.2.4.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 7.2.

The DSA Mapper source code structure is depicted in Figure 28.

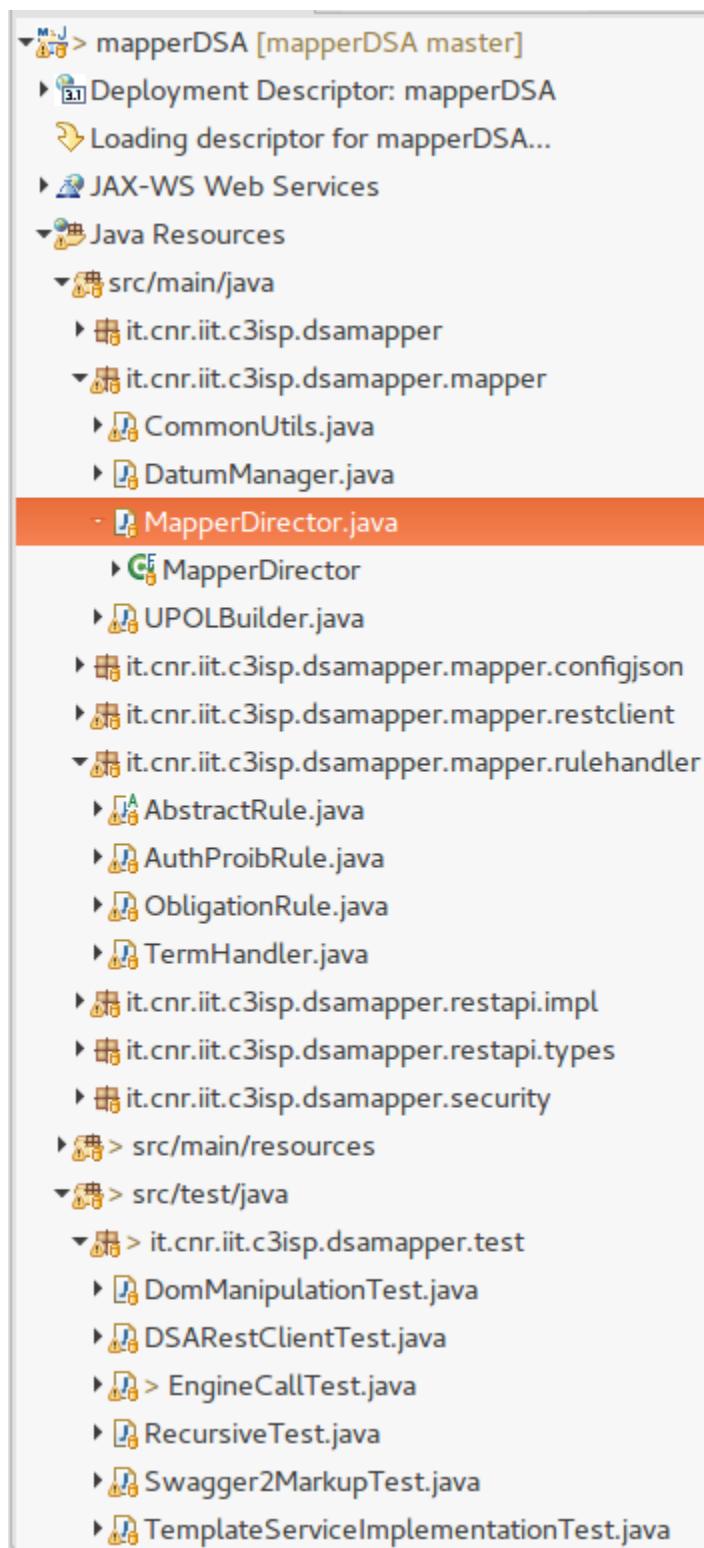


Figure 28. DSA Mapper Code Structure

The packages are logically divided for the purpose they must accomplish:

- **Dsamapper** contains all public methods for external calls.
- **Configjson** takes care to load the configuration for the cnl conversion
- **Restclient** contains the classes for calls remote web services (E.g. Upload a dsa after being mapped)

- **Rulehandler** contains abstract and derived classes to represent Authorization, Prohibition and Obligations
- **Restapi** has all implementation for the actual Web Service of the mapper
- **Tests** contains all the Test Cases.

5. Data collection and Usage Enforcement: The Information Sharing Infrastructure (ISI)

The Information Sharing Infrastructure is one of the foundations of the C3ISP architecture, responsible for data management and sharing. More precisely, besides basic CRUD operations on CTI data, it is devoted at providing sharing capabilities regulated through sophisticated policies, embedded in Data Sharing Agreements (DSAs);

5.1. Information Sharing Infrastructure API

5.1.1. Component description

The ISI API is a Java web application that is in charge of exposing a RESTful interface for the whole ISI. It is the entry point for ISI users.

The ISI API has two main objectives:

- To offer operations for DPO management
- To act as front-end for search functionalities

The component achieves its objectives by interacting with other components, acting as mediator between a user and the components responsible for the execution of the requested operation. It also ensures requestor's authentication through its interaction with the CSS, as well as some syntactic/ and sanity checks on input parameters.

The ISI API interacts:

- For DPO management operations: with DSA Adapter Front-End
- For DPO search operations, with DPOS
- For DSA search operations, with DSA Store

The ISI API is implemented using the Spring framework family. It relies on the latter for authentication, syntax and sanity checks. Simple input transformation (when necessary) are performed in order to create the correct calls to other C3ISP components.

5.1.2. Maturation status

The ISI API was not part of the set of assets considered for D8.1 but it is deemed necessary in order to offer a consistent and unique entry point to users, where security mechanisms for authentication and input sanitisation may be safely enforced.

At M24, the ISI API is implemented and supports most of the available C3ISP functionalities at their maturation level. Being a component in charge of exposing other component's functionalities, it is not foreseen to be significantly changed before the end of the project but naturally, it will be updated as the set of C3ISP functionalities will grow.

5.1.3. Requirement Analysis at M24

The ISI API was not part of components analysed in D8.1, therefore this section will contain a list of requirements for the component to be fulfilled by the end of the project.

ID	Priority	Requirement
----	----------	-------------

C3ISP-Com-ISI-API-001	MUST	The component must be able to offer the C3ISP ISI functionalities.
C3ISP-Com-ISI-API-002	MUST	The component must ensure user authentication.
C3ISP-Com-ISI-API-003	MUST	The component must ensure input sanitisation.

The requirement analysis may be detailed as follows:

ID	MET	Description
C3ISP-Com-ISI-API-001	PARTIALLY	The current version of the ISI API offers support for ISI functionalities, but it has to be updated: <ul style="list-style-type: none"> - When new functionalities will be made available or updated - With respect to the “move” operation, to support DPO transfers from Local to Central ISI (see D7.3)
C3ISP-Com-ISI-API-002	YES	The current version of the ISI API provides support for user authentication.
C3ISP-Com-ISI-API-003	YES	The current version of the ISI API provides support for data sanitisation using standard Spring functionalities (type checking, data serialization and de-serialization).

5.1.4. First release of the component

The first release of the ISI API runs as a Java application powered by the Spring framework. It exposes a RESTful API by means of a number of classes, one per method, as detailed in the following Table 3. Each of such classes performs checks on the input parameters (JSON validation against the model definition, sanity checks...), to create a new call to other ISI components in charge of the invoked functionality.

Table 3: ISI API RESTful methods

Method Name	Note	Parameter Example
<i>/v1/dpo</i>	A POST request to the URL with the parameters as for the example will trigger the DPO create workflow. The return element of the call is a	HTTP Form with: <ul style="list-style-type: none"> - fileToSubmit: input file - inputMetadata:

	<p>DPO_id for the submitted file, if call is successful.</p>	<pre>{ "Request" : { "Attribute" : [{ "AttributeId" : "ns:c3isp:dpo- metadata", "Value" : "{\"id\": \"4000123\", \"dsa_id\": \" DSA-56976731-3c16-46cc-a4e1- 8384c6208eb0\", \"start_time\": \"2 017-12- 14T12:00:00.0Z\", \"end_time\": \" 2017-12- 14T18:01:01.0Z\", \"event_type\": \"Firewall Event\", \"organization\": \"3DRep o\"}]", "DataType" : "string" }] } }</pre>
<p>/v1/dpo/<dpo_id></p>	<p>A GET request to this endpoint (once user is authenticated), if authorised, will return the DPO associated to the DPO_id. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell.</p>	<p>HTTP Header x-c3isp-input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p>/v1/dpo/<dpo_id></p>	<p>A DELETE request to this endpoint (once user is authenticated), if authorised, will delete the DPO associated to the DPO_id. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell.</p>	<p>HTTP Header x-c3isp-input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, </pre>

		<pre>"Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p><i>/v1/move/dpo/<dpo_id></i></p>	<p>A POST request to this endpoint will trigger a move operation from a Local ISI to a Central ISI. Encryption parameters and address of Central ISI must be configured by an administrator prior to the call. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell.</p>	<p>HTTP Header x-c3isp-input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p><i>/v1/prepareData/</i></p>	<p>A POST request to this endpoint will trigger the execution of a workflow for the creation of a Data Lake Buffer, populated with the specified DPO_id(s). See section 5.3 for more details. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell.</p>	<p><set of Buffer Manager parameters, see section Erreur ! Source du renvoi introuvable.></p> <p>HTTP Header x-c3isp-input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p><i>/v1/search/<store>/<longResultFlag></i></p>	<p>A POST request to this endpoint will trigger the execution of a search operation on DPOS or DSA Store. See section Erreur ! Source du renvoi introuvable. for more details. It is possible to pass the desired query string using the metadata</p>	<p>HTTP Form input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" :</pre>

	format as specified in the next cell.	<pre> "ns:c3isp:search-string", "Value" : <DPOS search string as in section Erreur ! Source du renvoi introuvable.>, "DataType" : "string" }] } } </pre>
--	---------------------------------------	---

5.1.4.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/isiApi/isi-api>

5.1.4.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 3.4.1.

The source code of ISI API is structured as depicted in Figure 29.

The main package *eu.c3isp.isi.api* contains the *ApplicationDeployer* class, in charge of enabling the Spring framework activation for the web application.

The package *eu.c3isp.isi.api.restapi.impl* contains the implementation of all exposed APIs, one class per method.

The package *eu.c3isp.isi.api.restapi.types* has all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

Package *eu.c3isp.isi.api.restapi.types.xacml* contains some utility classes and metadata object containers to allow a transformation from C3ISP JSON objects to XACML JSON objects, used for easing the authorization checks triggered by DSA Adapter Front-End.

Lastly, *eu.c3isp.isi.api.restapi.types.security* has the Spring security initialization.

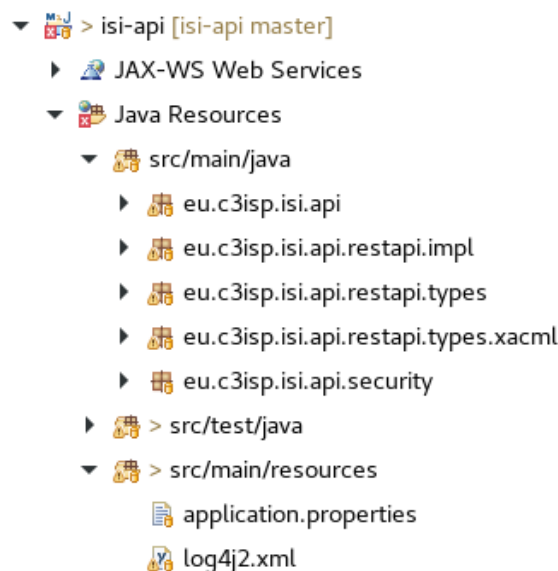


Figure 29: The ISI Api source code packages from the Eclipse IDE

5.2. *Data Protected Object Storage (DPOS)*

5.2.1. **Component description**

The **Data Protected Object Storage (DPOS)** persistently stores the CTI data provided by the Prosumers in the form of the C3ISP data bundle (or DPO – Data Protected Object).

With respect to the storage of CTI data, the DPOS implements Create, Read and Delete functionality. Since each DPO is protected, and therefore immutable, no Update operation is appropriate in this context.

Each DPO is identified by a unique DPO ID, provided by the client. Within the DPOS, the DPO is implemented as four separate records:

- a CTI file,
- the associated **Data Sharing Agreement (DSA)** which protects the DSA by identifying authorizations, obligations and prohibitions associated with the CTI file (see Section 4)
- a **hash signature** of both the CTI file and the DSA agreement (see Section 5.10)
- a set of **metadata** key-value pairs, which provide both description and a searchable vocabulary.

As stated above, the DPOS provides a CRD + search functionality. It implements the following API methods:

- *CreateDPO*: Given a CTI file, and the associated DSA file, hash code and DPO metadata document, create a DPO in the DPOS.
- *ReadDPO*: Retrieve the four components of a DPO from the DPOS repository, given its DPO ID.
- *DeleteDPO*: Given a DPO ID, delete the corresponding DPO from the repository.
- *SearchDPO*: Given a JSON-based search string (see Section 5.2.4.2), query the DPO metadata repository, and return a set of metadata entries corresponding to the matching DPOs. This method returns either a set of DPO IDs or a set of full DPO metadata entries, based on the boolean *longResultFlag* parameter.

Like most other C3ISP Framework components, the DPOS is implemented using the Java SpringBoot Framework. The DPOS software provides a format and error-checking wrapper to both its storage and search backends, parses queries and translates it to the format supported by the search backend and exposes the DPOS API via a REST interface.

5.2.2. **Maturation status**

At M24, the prototype is fully functional, and integrated with its storage backend. Its REST API is available at <https://isic3isp.iit.cnr.it/dpos-api/>.

In addition to the storage functionality described in the design phase (see deliverable D8.1), the DPOS additionally supports a search feature, which allows search on a set of metadata fields stored as part of the DPOS. The search feature allows for easier retrieval of data (for example, allowing a user to retrieve all data belonging to the parent organization without having to store an index of created data), and most importantly, allows collaborative analytics. Both the metadata fields and the query languages use a JSON-based format.

No target Technology Readiness Level was defined for the DPOS during the requirements-gathering phase in deliverable D8.1, since this component did not feature in the deliverable. Thanks to its successful integration with other ISI components, it currently achieves

approximately TRL 5, and in line with other C3ISP components, we hope to achieve TRL 6 by the end of the first validation cycle for both its CRD and search functionality.

5.2.3. Requirement Analysis at M24

Postponed

5.2.4. First release of the component

The current version of the DPOS uses two back-end technologies: a storage backend, and a search backend. The storage backend stores the CTI data itself, the DSA, and the hash signature on a filesystem, while the search backend stores, indexes, and provides the search interface to DPO metadata provided by the DPOS client.

5.2.4.1. DPO storage backend

The DPOS supports several storage backend alternatives, depending on the user's need for either simplicity of installation or large capacity. The choice of implementation is determined by activating the corresponding profile in the DPOS configuration file, `application.properties`:

```
#centraldpos <- large-capacity configuration, such as in a central ISI
spring.profiles.active=centraldpos
#spring.data.mongodb.uri=mongodb://dpostoremgr:dpostoremgr-
kent@iaic3isp.iit.cnr.it:27017/dpostore
#spring.hadoop.config.fs.defaultFS=hdfs://iaic3isp.iit.cnr.it:19000

# localdpos <- small-scale configuration, such as on a Local ISI node
spring.profiles.active=localdpos
spring.data.mongodb.uri=mongodb://localhost:27017/dpostore
spring.hadoop.config.fs.defaultFS=hdfs://localhost:19000
```

The large capacity version of the back-end will support a DPOS deployment requiring high storage capacity and high performance, for example in a central ISI installation. This version is implemented using the Hadoop filesystem (HDFS).

The scaled-down version of the back end is implemented on a standard filesystem. This version may be used in lower-capacity DPOS installations, such as the one used by a Local ISI.

The physical structure of the DPO mirrors its logical structure. The CTI data, the DSA, and the hash are stored as filesystem files. The filename is constructed from the DPO ID, which marks the membership of the file in a DPO, and an extension, which marks the purpose of the file.

Purpose	Filename	Example
CTI payload	dpo_id.payload	66c33d30-f086-4409-9b0c-bcca431f2006.payload
Hash signature	dpo_id.sign	66c33d30-f086-4409-9b0c-bcca431f2006.sign
DSA file	dpo_id.dsa	66c33d30-f086-4409-9b0c-bcca431f2006.dsa

5.2.4.2. DPO search backend

The DPO search backend is implemented by MongoDB NoSQL database⁹ and stores metadata included with the C3ISP DPO. This metadata is used by the C3ISP Framework to classify and search CTI data to enable collaborative analytics.

This searchable metadata is packaged with the DPO, but unlike the DSA-protected CTI data, it may be indexed outside of the DPO. The metadata includes CTI metadata, as well as DPO-specific metadata, also known as DPO (data-protected object) metadata and will form part of the CTI Bundle header. Like its associated ontology and query format, it uses JSON format.

Vocabulary

The DPO metadata is described by a simple ontology, describing the metadata vocabulary, data type, and operators allowed in the queries. It contains one compulsory *id* field, which uniquely identifies the DPO within the DPOS. The ontology must define the *id* field, as well as the other administrator-defined metadata fields.

The metadata is divided into two types:

- CTI metadata: the metadata specific to the CTI data being stored in the DPO. This metadata can be inferred directly from the CTI data.
- Examples: *start_time*, *end_time*, *event_type*
- All other metadata: additional metadata added by the Bundle Manager and DPOS that describe attributes specific to C3ISP network
- Examples: *id*, *dsa_id*

CTI metadata may change during CTI Bundle creation if its associated CTI data is changed by the pre-bundling DMOs specified in the DSA. Once the CTI Bundle is stored in the DPOS, the metadata will remain static for the lifetime of the data-protected CTI Bundle (DPO).

The ontology will be structured as follows:

```
{
  "name": "Name of schema",
  "attributes": [
    {
      "name": "Attribute1",
      "operators": ["op1", .. "opK"],
      "valueType": "Type1",
      "constraint": <valueType-specific constraint, such as length upper-
and-lowerbounds>
    },
    ...
    {
      "name": "AttributeN",
      "operators": ["opN1",..., opNK"],
      "valueType": "TypeN"
    }
  ]
}'
```

where permitted operators are in [eq, gt, gte, in, lt, lte, ne, nin], and where the valueTypes are in [string, number, boolean, date, array, enumerated]

Example:

```
{
  "name": "DPO Metadata",
```

⁹ MongoDB Server, available: <https://www.mongodb.com/>

```

    "attributes":[
    {
        "name":"id",
        "operators":["eq"],
        "valueType":"string",
        "constraints":[1,256]
    },
    {
        "name":"dsa_id",
        "operators":["eq"],
        "valueType":"string",
        "constraints":[1,256]
    },
    {
        "name":"start_time",
        "operators":["Le","lte","gt","gte","eq","ne"],
        "valueType":"date",
        "constraints": ["2018-12-16T14:00:00.0Z","2018-12-20T14:00:00.0Z"]
    },
    {
        "name":"event_type",
        "operators":["eq","ne"],
        "valueType":"string"
    },
    {
        "name":"organization",
        "operators":["eq","ne"],
        "valueType":"string"
    },
    {
        "name":"severity_level",
        "operators":["eq","ne"],
        "valueType":"enumerated",
        "constraints": ["INFORMATIONAL", "WARNING", "MINOR", "MAJOR",
"CRITICAL"]
    }
    ]
}

```

Format

DPO metadata will be stored in JSONformat, in a simple flat structure:

```

{
    "Attribute1": "Value1",
    "Attribute2": "Value2",
    ...
    "AttributeN": "ValueN"
}

```

where the attributes and value types are defined in the metadata ontology.

Example:

```

{
    "id":"12345",
    "dsa_id" : "54321",
    "start_time" : "2017-12-14T12:00:00.0Z",
    "end_time" : "2017-12-14T18:01:01.0Z",
    "event_type" : "Firewall Event",
    "organization" : "3DRepo"
}

```

Note: all input dates for metadata and queries must be in ISO 8601 format. They are stored zoned in UTC.

Search

Once DPO metadata is stored within the DPOS, the search backend facilitates clients to query it by passing a search string to the *SearchDPO* REST call. The search string is a JSON document with the same ontology as the DPO metadata document, containing a flat list of criteria, which include DPO metadata fields with operators and query values. The search string also specifies a combining rule (“and” or “or”) to combine the query criteria. The DPOS uses this search string to perform simple, field-specific matching on the metadata. For example, search for *event_type* will be a single string comparison, while search on time range may be an overlap comparison.

Format:

```
search_string= '{
  "combining_rule": "rule",
  "criteria": [
    {
      "attribute": "Attribute1",
      "operator": "op1",
      "value": "QueryValue1"
    }
    ...
    {
      "attribute": "AttributeN",
      "operator": "opK"
      "value": "QueryValueN"
    }
  ]
}'
```

where $rule \in [\text{and}, \text{or}]$, $Attribute1..AttributeN$ and $op1..opK$ are defined in the ontology file for this operator, and $QueryValue1..QueryValueN$ correspond to the type defined in the metadata ontology.

Example:

```
search_string = '{
  "combining_rule": "and",
  "criteria": [
    {
      "attribute": "event_type",
      "operator": "eq",
      "value": "Firewall Event"
    },
    {
      "attribute": "start_time",
      "operator": "gt",
      "value": "2017-12-12T12:00:00.0Z"
    },
    {
      "attribute": "end_time",
      "operator": "lt",
```

```
        "value": "2017-12-12T18:00:00.0Z"  
    }  
  ]  
}'
```

The above query searches for all DPOs which contain CTIs of type Firewall Event that contain records of events observed between noon and 6pm on 12 December 2017. When this search is performed by the C3ISP analytics service, it may allow collaborative analytics to be performed on matching data from multiple organizations.

5.2.4.3. *Link to Source Code*

The GIT repository for the DPOS is hosted by CNR, at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp5/DPOS/dpos-api>. The source code tree is shared with the DPOS API.

5.2.4.4. *Source Code Description*

The code structure of the DPOS and DPOS API is depicted in Figure 30. Please note the two alternative DPOS implementations: DPOSCentral and DPOSLocal, both implementing the DPOSInterface. The choice of backend implementation is configured by a parameter in the application.properties file.

The two main object types are DPOMetadata and DPO, representing respectively the DPO metadata header and the remaining three components of the DPO. Their SpringBoot repositories are then implemented by DPOMetadataRepository and DPOSInterface, which then interface with the MongoDB search backend and either HDFS (DPOSCentral) or local filesystem (DPOSLocal) storage backend, depending on the profile selected in application.properties resource file. The coordination between the two backends is implemented by the DPOS class, which treats each DPO as a single unit.

The DPO metadata ontology is defined in the dpo-metadata-ontology.json resource file, and read into the DPOMetadataOntology class. This ontology defines DPO metadata fields allowable in the DPO metadataheader, and also which fields are valid for use in DPO queries. The DPO query files manages parsing and validating DPO queries against the configured ontology, and the translation of queries into a format usable by the backend repository.

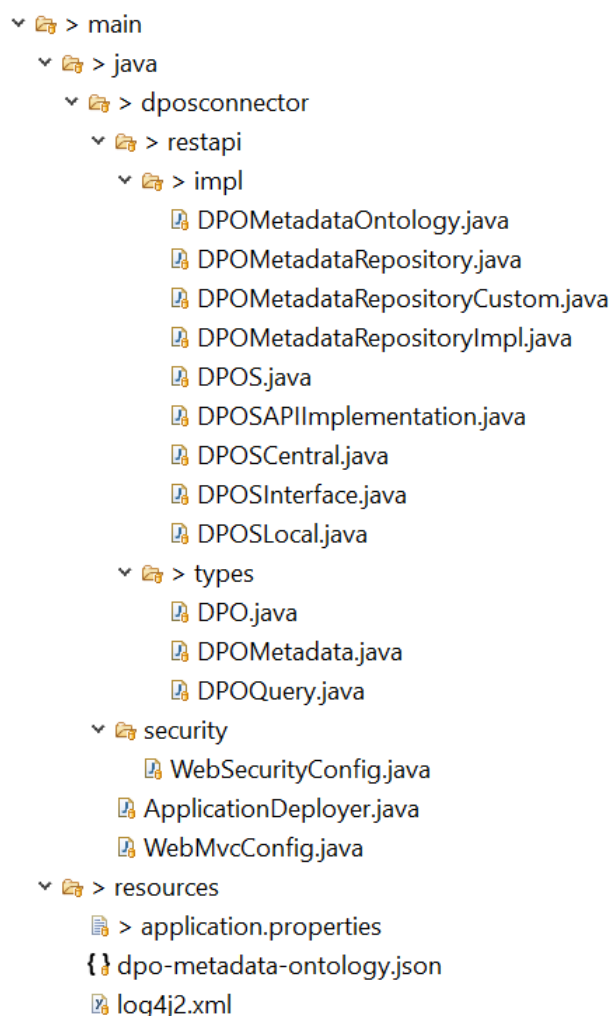


Figure 30: DPOS and DPOS API code structure

5.3. Buffer –Manager

5.3.1. Component description

The **Buffer Manager** is a component of the C3ISP Framework, introduced with the goal of granting temporary access to security data that are required by C3ISP-aware Analytics services and Legacy Analytics services. It operates by transferring the DPOs that are required by the Analytic service to a temporary storage, called a Data Lake, which the analytic itself will have direct access to. Data are manipulated according to existing DSAs before being written to the temporary storage, thus making sure that the Analytic services only have access to the data they have right to read.

A **Data Lake** is a common interface introduced with the Buffer Manager and it's used for storing and retrieving security data; it can be implemented on a variety of storage types, from databases to distributed file systems. Data Lakes can also be of two distinct types: *DataLakeBuffer* (DLB) and *VirtualDataLake* (VDL) that are different in terms of how they access and read the data; different implementations of Data Lakes for different storages define the differences between those two types.

Data Lakes instances can be created, populated and deleted (released) using the **Buffer Manager API** along with a URI that is provided upon creation, which at any moment is unique among all the active Data Lake URIs.

The Analytics services are able to autonomously connect to any Data Lake instance using the URI and the right protocol, without involving the Buffer Manager in the process. This requires an authentication system to be set when exposing the storage, in order to prevent unauthorized access to security. Generally speaking, authenticating access to Data Lakes is not a task of the Buffer Manager.

Three implementations of the Data Lake interface are planned to be released in the final version, each wrapping a different storage type: traditional *filesystem* (Linux and Windows), a *MySQL* instance and a distributed file system running on *ApacheHDFS*. The system is designed to ease the creation of new Data Lake implementations for different storage types in the future.

In the current design the Buffer Manager runs in the ISI virtual machine and is exposed by the ISI API.

5.3.2. Maturation status

The Buffer Manager is a new component of the C3ISP infrastructure.

5.3.3. Requirement Analysis at M24

postponed

5.3.4. First release of the component

The Y2 release of the Buffer Manager implements the following tasks:

- Create new Data Lake instances
- Produce a URI that uniquely identifies any instance of a Data Lake among the others. The URI can be used to access that instance directly with read and write privileges
- Fetch DPOs from the central ISI using the ISI API and write them to a Data Lake
- Write new data to an existing Data Lake instance. Data and access credentials (when needed) are not stored by the Buffer Manager and must be provided by the requestor.
- Delete an instance of Data Lake using the URI. Access credentials (when needed) are not stored by the Buffer Manager and must be provided by the requestor.

In particular, the URIs are needed in order to read/write data from the related instance of Data Lake and to release it, thus they must be unique. The URI can be used to access the storage using the appropriate protocol, e.g.: after creating an instance of a MySQL Data Lake implementation, a URI that starts with `jdbc:mysql://` will be returned by the BufferManager API, so it will be possible to access the MySQL database using JDBC.

As of M24 there are two use cases that involve the Buffer Manager. In the first use case, Analytics services call the Buffer Manager indirectly as a consequence of the call to the ISI

API `/prepareData` endpoint. Analytics send a list of DPOs they need to run the analysis, then the Buffer Manager fetches the data using the ISI API, writes them into a Data Lake and calls the **Format Adapter** component that transform the data to the right format (e.g. CSV or CEF) as requested by the Analytic service. Finally, the URI of the new Data Lake is returned to the requesting service.

In the second use case, Data Lakes are used as a temporary storage by the **DMO Engine** during the manipulation of DPO. Whenever a DPO is requested through the ISI API (e.g. as part of the first use case), it must be read from the DPOS and manipulated in order to comply with one or more DSAs. A Data Lake is requested by the DMO Engine directly to the Buffer Manager in order to store the DPO, then the Data Lake URI is processed by a DMO Operation and finally returned by the ISI API.

Both use cases end with the release of the Data Lake and the deletion of all data that was ever written in it.

The Buffer Manager component is developed using **Spring Boot** and runs in the ISI environment. In addition to the Buffer Manager API the application exposes a Swagger-UI definition that contains extended documentation about the functions of the API, which are:

- A `/prepareData` endpoint that creates a Data Lake instance and populates it with data read from the ISI API.
 - *Parameters:* the Data Lake type (VDL or DLB), the storage type of the Data Lake, the ISI API metadata, the desired output format of the data, the name of the Analytic service that is requesting the operation and the list of required DPOs.
 - *Return value:* The URI of the new Data Lake instance.
- A `/prepareEmptyDataLake` endpoint that creates an empty Data Lake instance.
 - *Parameters:* Data Lake type and storage type
 - *Return value:* The URI of the new Data Lake instance.
- A `/populateDataLake` endpoint that writes data to a new file inside an existing Data Lake. This function does not read the data from the ISI: it populates the Data Lake with the content provided as a parameter.
 - *Parameters:* the URI of an existing Data Lake instance and the content of the new file. Optionally, the name of the file where the data will be saved.
 - *Return value:* a message with the outcome of the write operation.
- A `/releaseDataLake` endpoint that can be used to delete any Data Lake by providing its URI and the access credentials.
 - *Parameters:* The Data Lake URI (must contain access credentials, when required by the protocol used to access the Data Lake)
 - *Return value:* a message with the outcome of the release operation.

Also, the Data Lake interface has been implemented and is fully functional for two storage types out of three planned:

- a *File System Data Lake* that uses a traditional file system to store the data. Data Lakes are created in a dedicated folder that is then exposed using NFS. URIs follow this model:

```
file:///opt/isi/datalakebuffer/<data_lake_name>
```

- A *MySQL Data Lake*, built for analytic services that need to read data from a relational DBMS. This implementation provides a MySQL URL. Credentials are provided as properties in the URL in order to grant access only to the rightful owner:

```
jdbc:mysql://iaic3isp.iit.cnr.it:3306/<data_lake_name>?usr=<user>&
psw=<password>
```

Future versions of the Buffer Manager will include a new Data Lake implementation for the Apache *Hadoop Distributed File System* (HDFS), with the goal of integrating Hadoop tools for big-data analysis into C3ISP Analytics services.

5.3.4.1. Link to Source Code

The code is hosted on CNR servers, at <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/buffer-manager/springswagger-template>.

5.3.4.2. Source Code Description

Signature of the APIs and their description is reported in the previous pages. The structure of the source code of the Buffer Manager is shown in the image below:

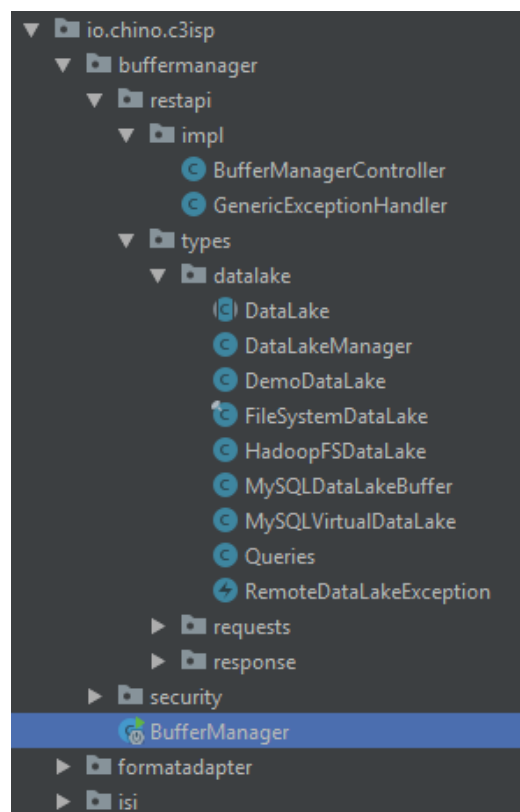


Figure 31: Buffer Manager code structure

The component is implemented as a **Spring Boot Application**, whose entry point is class `BufferManager`; the application's code is developed in package `io.chino.c3isp` and its subpackages. From now on we will refer to the root package as '*'.

Here is an overview of the main classes and packages of this component:

- The **Buffer Manager API** definition is contained inside the `*.buffermanager.restapi.impl.BufferManagerController` class; here is implemented the logic of the API calls. The definition is parsed using *swagger-annotation* which produces a web page with the API documentation. This page can be found by navigating to `/swagger-ui.html`.
- The package `*.buffermanager.restapi.types` contains code that is required as part of the Buffer Manager's logic. Classes in `requests` and `responses` are used for serialization / deserialization of API requests and responses – mapping is handled by Spring Boot, which converts JSON objects to POJO and vice-versa.

Package `datalakes` contains the **Data Lake interface** (DL) as well as all the planned implementations (MySQL, file system and HDFS) and some utility classes. The `DataLakeManager` is in charge of creating instances of DL; it currently supports MySQL and file system instances, while the `HadoopFSDataLake` is a work in progress and is not considered by the DL manager. `DemoDataLake` is only used for testing and as a placeholder for inactive DLs (such as the HDFS implementation). It only returns stub responses and does not affect a particular storage.

Class `Queries` contains the SQL statements that are used by the `MySQLDataLake` implementation.

- Inside `*.buffermanager.security` is placed the configuration file of **Spring Boot WebSecurity**, where are specified the URLs to be protected by authentication. The Swagger resources are unprotected, while API are secured with Basic authentication (username / password).
- Finally, the packages `*.formatadapter` and `*.isi` contain the client-side code of the Buffer Manager, i.e. the API clients that interact with other C3ISP Framework components. Both clients rely on the *OkHttp3* library to communicate with other components.

The **ISI API client** is fully tested and active: it can retrieve DPOs as part of the workflow of `/prepareData` endpoint.

The **Format Adapter API client** is still in a testing phase and is not used as part of the current workflow.

In conclusion, we provide a list of the dependencies which are required by the Buffer Manager. The component uses Maven for dependency management.

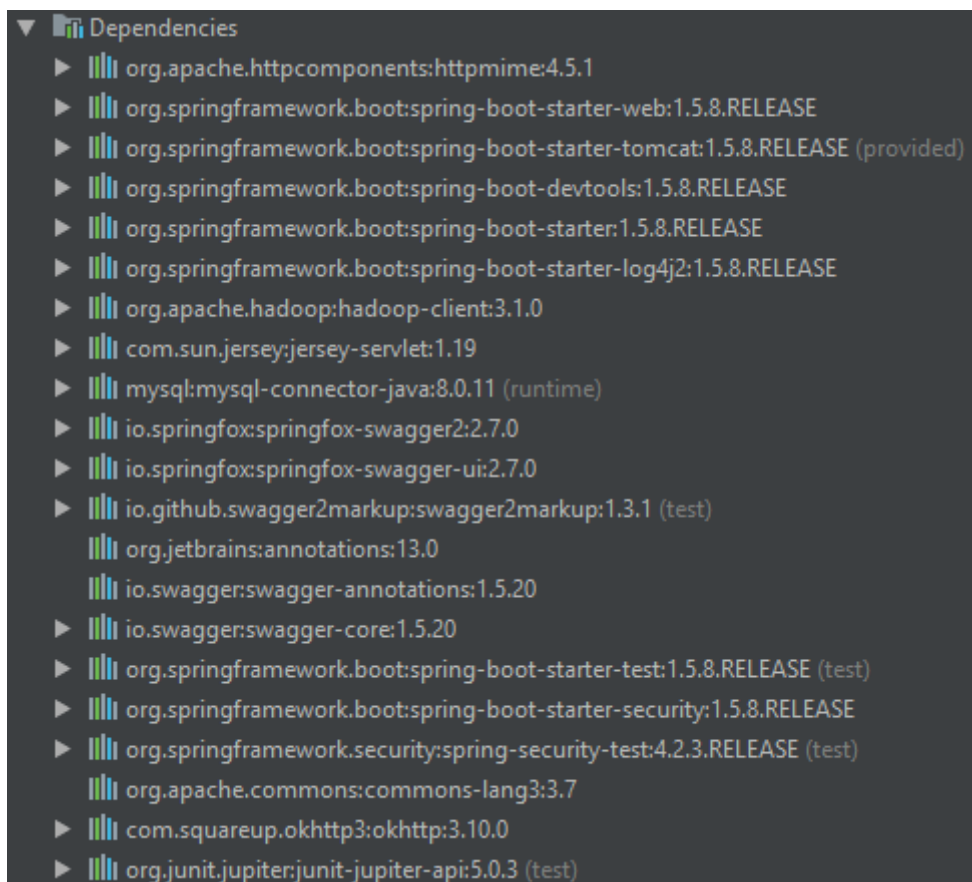


Figure 32: Dependencies required by the BM

5.4. Format Adapter

5.4.1. Component description

The **Format Adapter** is the component of the C3ISP Framework which adapts the format of CTI data to a STIX standard format to be easily processed by the various C3ISP components. The Format Adapter is able to detect the actual format of the data and convert it automatically to the desired format. The component is developed as an API with the OpenAPI specifications which are implemented thanks to Swagger.

5.4.2. Maturation status

The document “Format Adapter Details” describes all the CTI data to be converted and their right format.

CTI data	Integration Status
Monitoring of connections to malicious hosts	-
Monitoring of Domain Generation Algorithm DNS-request	-
Email Analysis	-
Firewall Schema	-
Anti-Malware Schema	-
Security Report Sharing	-

Enterprise Pilot (Intrusion Detection Events, Malware Events, Network Traffic Events, Web Events)	X
---	---

5.4.3. Requirement Analysis at M24

postponed

5.4.4. First release of the component

The Format Adapter can be reached through this URL: <https://isic3isp.iit.cnr.it:9443/format-adapter/api-docs/>.

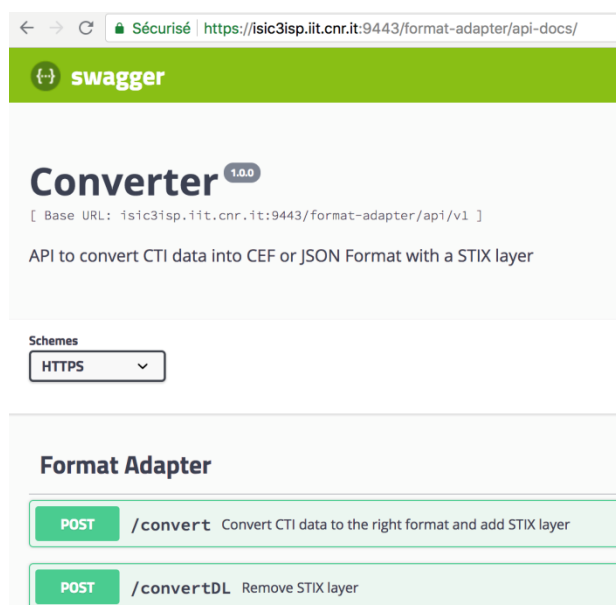


Figure 33: Screenshot of Format Adapter Swagger UI

This current release of the Format Adapter automatically detects the CTI format and translates it into the expected output format (CEF, JSON) embedded in a STIX layer. It can also remove the STIX layer when needed.

The UI of the Format Adapter enables to upload file and to retrieve the formatted content from the response body of the request. An example of CTI content, CURL request and response can be found below:

- CTI content

```
15-Sep-2017 16:11:43.431 client 192.168.1.2#37239 (www.google.com): query: www.google.com IN A -EDC (192.168.1.9)
15-Sep-2017 16:11:44.474 client 192.168.1.3#57203 (www.rai.it): query: www.rai.it IN A + (192.168.1.9)
```

- CURL request

```
curl -X POST "https://isic3isp.iit.cnr.it:9443/format-adapter/api/v1/convert" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F file=@DNS_Vendor_DNS_CED_1.0_100_DNSquery_5_.txt;type=text/plain
```

- Response body

```
{
  "spec_version": "2.0",
  "type": "stix-bundle",
  "id": "stix-bundle--76b7b52f3b67c753b05eb4ed17a95573445250ce",
}
```

```

"objects": [
  {
    "type": "observed-data",
    "id": "observed-data--ed54675fba43d3039af4f014da8c23a391901501",
    "created": "2018-09-02T22:02:22.789Z",
    "modified": "2018-09-02T22:02:22.789Z",
    "first_observed": "2018-09-02T22:02:22.789Z",
    "last_observed": "2018-09-02T22:02:22.789Z",
    "cybox": {
      "spec_version": "3.0",
      "objects": [
        {
          "type": "array",
          "minitems": "1",
          "items": [
            [
              "CEF:0|DNS_Vendor|DNS_CED|1.0|100|DNSquery|5|src=192.168.1.2 spt=37239 msg=INA-EDC(192.168.1.9) end=1505484703431 dtz=Europe/Berlin",
              "CEF:0|DNS_Vendor|DNS_CED|1.0|100|DNSquery|5|src=192.168.1.3 spt=57203 msg=INA+(192.168.1.9) end=1505484704474 dtz=Europe/Berlin"
            ]
          ]
        }
      ]
    }
  ]
}

```

5.4.4.1. *Link to Source Code*

The source code is available at <https://isic3isp.iit.cnr.it:9443/format-adapter/api-docs/>

5.4.4.2. *Source Code Description*

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.2.

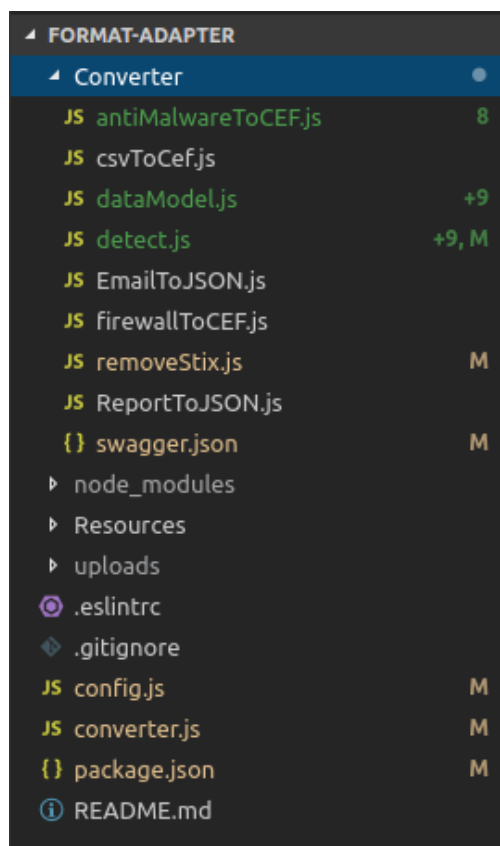


Figure 34: Format Adapter API code structure

The module is implemented as a NodeJS API whose entry point is the script `converter.js`. Below is an overview of the main files and folders of this component.

The folder **Converter** regroups all the scripts implementing the different format converters:

- `antiMalwareToCEF.js`: Convert the schema of the Anti-Malware CTI data to STIX CEF,
- `csvToCef.js`: Converter the csv file containing the log coming from a router and the request generated with the BIND DNS to STIX CEF
- `dataModel.js`: convert the Enterprise Pilot Data Models to STIX CEF
- `detect.js`: Detect what the original format of the file and choose the correct format adapter to use
- `EmailToJSON.js`: Convert email at the eml format to a STIX JSON
- `FirewallToCEF.js`: Convert the schema of the Firewall CTI data to STIX CEF
- `removeStix.js`: remove the STIX of a file already converted
- `ReportToJSON.js`: Convert reports coming from the scan software provided by Registro.it to STIX JSON
- `Swagger.json`: Contains swagger UI configuration (hosts, paths, parameters etc)

The folder **node_modules** contains the dependencies that has been installed.

The folder **Resources** contains example files that can be used to test the format adapter,

The **uploads** folder contains temporary files or data needed during the format adaptation,

The file `.eslintrc` is a configuration file for eslint, a pluggable and configurable linter tool for identifying and reporting on patterns in JavaScript. It helps maintain the quality of quality with ease.

`Config.js`: define the hostname, port and scheme.

`Package.json`: file with all the dependencies that need to be installed to run the module

5.5. *DSA Adapter Frontend*

5.5.1. **Component description**

The DSA Adapter Front-End is the entry point of the DSA Adapter subsystem. Due to the event-driven internal architecture of the DSA Adapter, that is centred on the Event Manager, it is necessary to offer a REST interface that can be easily consumed by other C3ISP components and in particular by the ISI API. It is also in charge of checking the authorizations for each requested operation, against:

- a global ISI authorization policy (configuration policy) and
- each individual DSAs for the requested resources.

Each policy is expressed using XACML and its extension, UPOL, developed under the umbrella of the C3ISP project.

The DSA Adapter Front-End is in charge of:

- Triggering the internal DSA Adapter operation workflows, by sending the appropriate event message to the Event Handler

- Trigger the authorization/usage control process for each involved resource, as implemented by the Continuous Authorization Engine and the Obligation Engine.

More in details, the DSA Adapter Front-End is integrated with:

- Continuous Authorization Engine, for the verification of access and usage control (continuous authorizations) directives
- Obligation Engine, for the enforcement of access and usage control obligations
- Bundle Manager, for the DPO management operations
- DMO Engine, for triggering data manipulation operations
- Event Handler, for exchanging messages with all other components

The DSA Adapter Front-End is implemented using the Spring framework family. It relies on the latter for exposing a REST interface, as well as for configuration management and dependency injection.

The DSA Adapter Front-End was not part of the set of assets considered for D8.1 but it is deemed necessary in order to offer a simplified and synchronous entry point to DSA Adapter functionalities. This choice allows for decoupling the event-driven paradigm implemented by the DSA Adapter from the rest of the ISI components, in order to simplify their interaction model.

At M24, the DSA Adapter Front-End is implemented and supports roughly the available C3ISP functionalities at their maturation level.

5.5.2. Requirement Analysis at M24

The ISI API was not part of components analysed in D8.1, therefore this section will contain a list of requirements for the component to be fulfilled by the end of the project.

ID	Priority	Requirement
C3ISP-Com-DSAAFE-001	MUST	The component must ensure the execution of all DSA Adapter operation workflows.
C3ISP-Com-DSAAFE -002	MUST	The component must ensure the verification of the access and usage control conditions for each requested operation.
C3ISP-Com-DSAAFE-003	MUST	The component must ensure the execution of data manipulation operations if requested by a DSA.

The requirement analysis may be detailed as follows:

ID	MET	Description
----	-----	-------------

<p>C3ISP-Com-DSAAFE-001</p>	<p>PARTIALLY</p>	<p>The current version of the DSA Adapter Front-End offers support for DSA Adapter functionalities, but it has to be updated:</p> <ul style="list-style-type: none"> - When new functionalities will be made available or updated - With respect to the “move” operation, to support DPO transfers from Local to Central ISI (see D7.3)
<p>C3ISP-Com-DSAAFE -002</p>	<p>PARTIALLY</p>	<p>The current version of the DSA Adapter Front-End provides support access control and usage control obligations, however only a limited support for continuous access control.</p>
<p>C3ISP-Com-DSAAFE-003</p>	<p>PARTIALLY</p>	<p>The current version of the DSA Adapter Front-End provides support for data manipulation operations, however it must be enhanced with a tighter integration with DMO Engine, especially in the create DPO workflow, before a DPO is created.</p>

5.5.3. First release of the component

The first release of the DSA Adapter Front-End runs as a Java application powered by the Spring framework. Its implementation relies on two main group of classes. One, the Event Listener, communicates with the Event Handler, sending and receiving messages towards and from the other DSA Adapter components. The other group is in charge of the implementation of the functionalities exposed to the ISI API. DSA Adapter Front-End exposes a RESTful API by means of a number of classes, as detailed in the following Table 4.

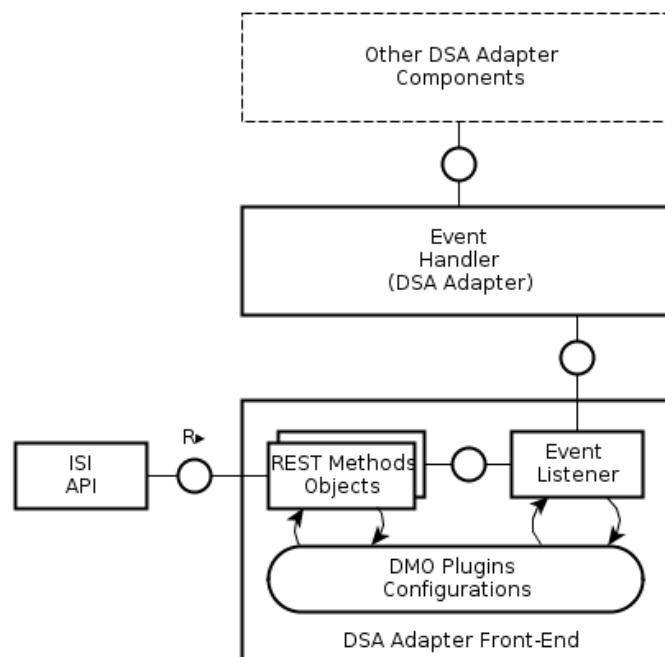


Figure 35: DSA Adapter Front-End fine-grained architecture.

Table 4: DSA Adapter Front-End RESTful methods.

Method Name	Note	Parameter Example
<i>/v1/create</i>	A POST request to the URL with the parameters as for the example will trigger the DPO create workflow. The return element of the call is a DPO_id for the submitted file, if call is successful. The implementation differs from ISI API in that the call, with the same parameters, triggers the creation of a message to the Bundle Manager for the creation of a new DPO (message type “bmc”).	<p>HTTP Form with:</p> <ul style="list-style-type: none"> - fileToSubmit: input file - inputMetadata: <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : "ns:c3isp:dpo- metadata", "Value" : "{\"id\": \"4000123\", \"dsa_id\": \" DSA-56976731-3c16-46cc-a4e1- 8384c6208eb0\", \"start_time\": \"2 017-12- 14T12:00:00.0Z\", \"end_time\": \" 2017-12- 14T18:01:01.0Z\", \"event_type\": \"Firewall Event\", \"organization\": \"3DRep o\"}]", "DataType" : "string" }] } }</pre>
<i>/v1/dpo/<dpo_id></i>	A GET request to this endpoint (once user is authenticated), if authorised, will return the DPO associated to the DPO_id. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell. The implementation differs from ISI API in that the call, with the same parameters, triggers the creation of a message to the Bundle Manager for getting a DPO with the specified DPO_id (message type “bmr”).	<p>HTTP Header x-c3isp- input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<i>/v1/dpo/<dpo_id></i>	A DELETE request to this endpoint (once user is authenticated), if	<p>HTTP Header x-c3isp-</p>

	<p>authorised, will delete the DPO associated to the DPO_id. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell. The implementation differs from ISI API in that the call, with the same parameters, triggers the creation of a message to the Bundle Manager for the deletion of a DPO with the specified DPO_id (message type “bmd”).</p>	<pre>input_metadata: { "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p><i>/v1/move/dpo/<dpo_id></i></p>	<p>A POST request to this endpoint will trigger a move operation from a Local ISI to a Central ISI. Encryption parameters and address of Central ISI must be configured by an administrator prior to the call. Optionally, it is possible to pass additional parameters to the call using the metadata format as specified in the next cell.</p>	<p>HTTP Header x-c3isp-input_metadata:</p> <pre>{ "Request" : { "Attribute" : [{ "AttributeId" : <any desired metadata>, "Value" : <metadata value>, "DataType" : "string" }] } }</pre>
<p><i>/v1/eventNotification /</i></p>	<p>A POST request to this endpoint will trigger the processing of an incoming message. The DSA API Front-End registers to a number of events/messages of interests (bundle manager create response, read response, delete response: respectively bmcr, bmrr, bmdr) and the Event Handler will call this method when one of such messages is received.</p>	<p>HTTP Form with element “event” of type:</p> <pre>{ "additionalProperties" : { "property1" : <any desired value>, "property2" : <any desired value> ...}, "eventType" : <bmcr bmrr bmdr>, "sessionId" : "string" }</pre>

5.5.3.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/dsa-adapter-frontend/dsa-adapter-frontend> (binary)

5.5.3.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.1.1.

The structure of the source code of the DSA Adapter Front-End can be depicted as in Figure 36.

The main package `eu.c3isp.dsa.adapter.frontend.restapi` contains the `ApplicationDeployer` class, in charge of enabling the Spring framework activation for the web application.

The package `eu.c3isp.dsa.adapter.frontend.restapi.impl` contains the definition and implementation of the Event Listener class, required for interacting with the Event Handler, plus some utility classes.

Package `eu.c3isp.dsa.adapter.frontend.restapi.impl.ops`, instead, contains the class definitions for all exposed REST methods, each of them exposed by one class.

Package `eu.c3isp.dsa.adapter.frontend.restapi.types` caters for all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

Package `eu.c3isp.dsa.adapter.frontend.restapi.xacml` contains some utility classes and metadata object containers to allow a transformation from C3ISP JSON objects to XACML JSON objects, used to create authorization check requests and interpret the associated responses, as coming from the Continuous Authorization Engine.

Lastly, `eu.c3isp.dsa.adapter.frontend.restapi.security` has the Spring security initialization.

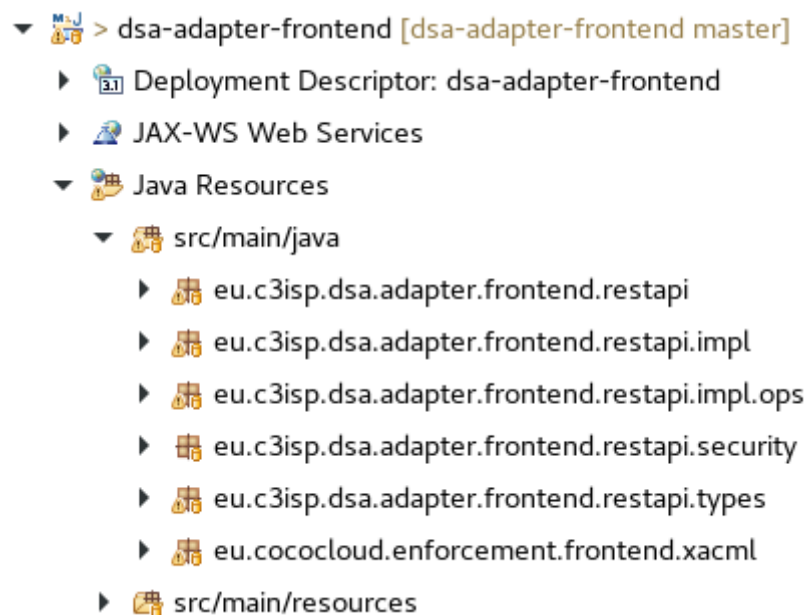


Figure 36: The DSA Adapter Front-End source code packages from the Eclipse IDE.

5.6. *Event Handler*

5.6.1. **Component description**

The Event Handler is a component in charge of dispatching messages (events) to its fellow DSA Adapter components. It implements a publish-subscribe message passing pattern. It is a Java web application that exposes a RESTful interface.

The importance of the Event Handler in the DSA Adapter architecture is to allow a transparent management of workflow and processes, to the benefit of the Usage Control components (Continuous Authorization Engine and Obligation Engine). Each member of the DSA Adapter has full visibility of the actual resources managed by the DSA Adapter and react accordingly, if needed. It is then possible, for example, for each of the two to interrupt the ongoing processing of a resource, in case the associated usage control directives (continuous conditions or usage control obligations) prescribe to do so.

The Event Handler is implemented using the Spring framework family. It relies on the latter for exposing a REST interface, as well as for configuration management and dependency injection.

5.6.2. **Maturation status**

The Event Handler was not part of the set of assets considered for D8.1 but it is deemed necessary in order to achieve transparency and to support the delivery of the DSA Adapter functionalities.

At M24, the DSA Adapter Front-End is implemented and it is deemed in a stable release. It currently supports the dispatching of events in an extensible message format, so that new event types can be freely supported without requiring any code modification. For these reasons, it is foreseen that no or minor adjustments to the implementation may come by the end of the project, while modifications and new message definitions may come as the C3ISP functionalities will extend.

5.6.3. **Requirement Analysis at M24**

The Event Handler was not part of components analysed in D8.1, therefore this section will contain a list of requirements for the component to be fulfilled by the end of the project.

ID	Priority	Requirement
C3ISP-Com-EH-001	MUST	The component must be able to support the exchange of all needed messages among DSA Adapter components.
C3ISP-Com-EH-002	MAY	DSA Adapter components must be able to subscribe and receive message of interest in “push” (i.e., Event Handler transmits to the component a message of interest as soon as it is made available) _or in “pull” mode (components periodically poll the Event Handler for messages of interest received since their last poll).

The requirement analysis may be detailed as follows:

ID	MET	Description
C3ISP-Com-EH-001	YES	The current version of the Event Handler offers full support to message exchanges in the DSA Adapter.
C3ISP-Com-EH-002	PARTIALLY	The current version of the Event Handler provides support for “push” interaction model, while there is only an initial support for “pull” model, essentially in the API definition.

5.6.4. First release of the component

The first release of the Event Handler runs as a Java application powered by the Spring framework. The core of the component is the Event Listener class: it exposes a method (“*notifyEvent*”) to notify events to DSA Adapter components that previously subscribed to that message/event type. The same class caters for a number of functionalities to manage the subscription of a component to the Event Handler and get the list of subscribed messages.

The Event Handler interface is secured with authentication and authorization features. In fact, as the central element of the DSA Adapter, it must be protected from misuses that may compromise the good enforcement of the DSA prescriptions.

The definition of message types at M24 is as follows.

Message ID	Main Involved Component	Description
tryaccess	Continous Authorization Engine, Obligation Engine	This message type is used to verify the authorization (access control) of a requested operation.
tryaccess_response	Continous Authorization Engine, Obligation Engine	This message type contains the evaluation of an associated tryaccess.
tryaccess_multi	Continous Authorization Engine, Obligation Engine	This message type permits to trigger a special access control verification for multiple resources, especially useful for complex analytics operations with multiple applicable policies.

tryaccess_multi_response	Continuous Authorization Engine, Obligation Engine	This message type contains the evaluation of an associated tryaccess_response.
startaccess	Continuous Authorization Engine, Obligation Engine	This message type informs the DSA Adapter component that a requested resource is now being processed by the requestor.
startaccess_response	Continuous Authorization Engine, Obligation Engine	This message type acknowledges the beginning of a processing session communication.
endaccess	Continuous Authorization Engine, Obligation Engine	This message type is used to notify the termination of a processing session for a resource.
endaccess_response	Continuous Authorization Engine, Obligation Engine	This message type is the acknowledgement of the previous endaccess message.
revoke	Continuous Authorization Engine, Obligation Engine	This message type interrupts a processing session previously authorized
bmc	Bundle Manager, DSA Adapter Front-End	This message type asks to create a new DPO through the Bundle Manager.
bmcr	Bundle Manager, DSA Adapter Front-End	This message type caters for the result of a previously requested bmc.
bmr	Bundle Manager, DSA Adapter Front-End	Message type used to demand the retrieval of a DPO, in clear text.

bmrr	Bundle Manager, DSA Adapter Front-End	This message type caters for the result of a previously requested bmr.
bmd	Bundle Manager, DSA Adapter Front-End	Message type used to demand the deletion of a DPO.
bmdr	Bundle Manager, DSA Adapter Front-End	This message type caters for the result of a previously requested bmc.
dmoe	DMO Engine, Obligation Engine	Message type used to start a Data Manipulation Operation.
dmoer	DMO Engine, Obligation Engine	This message type caters for the result of a previously requested dmoe.

Table 5: Event Handler RESTful methods.

Method Name	Note	Parameter Example
<i>/v1/events</i>	A GET request is used to retrieve messages of a specific message type, previously subscribed.	HTTP query string with: - subscriptionURL: URL used to subscribe to the service
<i>/v1/notifyEvent</i>	A POST request to this endpoint permits to notify an event (message) to all subscribers of that message type.	HTTP Body: { "additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "sessionId" : "string" } }
<i>/v1/subscribe</i>	A POST request to this endpoint permits to subscribe a URL to a message feed of a specified message	HTTP Body: {

	type. An additional parameter specifies a “pull” subscription, otherwise “push” model is selected per default.	<pre>"additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "URL" : "string" }</pre>
<i>/v1/unsubscribe</i>	A POST request to this endpoint will unsubscribe a URL from a message feed for a specified message type.	<pre>HTTP Body: { "additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "URL" : "string" } }</pre>
<i>/v1/subscribers</i>	A GET request to this endpoint will retrieve all subscribers for a specified message type.	<pre>HTTP query parameter: eventType: event type acronym</pre>

5.6.4.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/event-handler/event-handler>
(binary)

5.6.4.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.1.2.

The main package *eu.c3isp.isi.dsaadapter.eventhandler.restapi* contains the `ApplicationDeployer` class, in charge of enabling the Spring framework activation for the web application.

The package *eu.c3isp.isi.dsaadapter.eventhandler.restapi.impl* contains the definition and implementation of the `Event Handler` class, that exposes all methods.

Package *eu.c3isp.isi.dsaadapter.eventhandler.restapi.types* caters for all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

Lastly, *eu.c3isp.isi.dsaadapter.eventhandler.restapi.security* has the Spring security initialization.

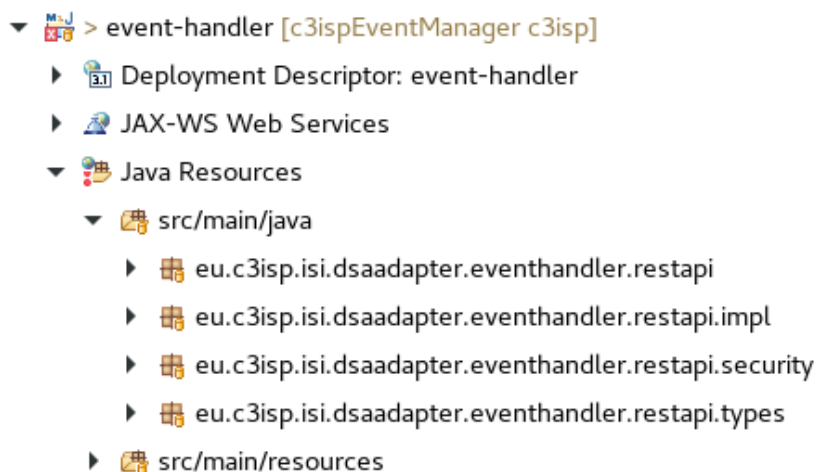


Figure 37: The Event Handler source code packages from the Eclipse IDE.

5.7. Continuous Authorization Engine

5.7.1. Component description

The **Continuous Authorization Engine (CAUTHENG)** is an authorization engine which allows the control of the usage of resources according to the Usage Control Model (UCON) defined in [20]. In particular, this component performs the authorization decision process both at access request time (such as in traditional access control models), to decide whether the access to the resource can be started, and continuously during the access time (i.e., while the resource is in use), to decide whether the access to the such resource can go on or must be terminated because of a policy violation or some countermeasures must be taken. The UCON model fits well those modern and dynamic scenarios, such as the ones of the C3ISP pilots, where resources, i.e., data, involved in computations that could last for a long time must be protected by security policies and where the access context could change while the usage of such data (or of the results derived from them) is still in progress. As a matter of fact, the UCON features are meant to guarantee that the right of a subject to use resources hold not only at access request time, but also while the usage is in progress.

This component has originally been developed by the Coco Cloud EU FP7 project (GA #610853), and within the C3ISP project is being matured in order to accommodate the features of the C3ISP scenario and of the related pilots, as will be described in the following.

In particular, at M24 the CAUTHENG component has been matured in order to be exploited in the DSA Adapter component of the ISI subsystem to enforce the usage control policies defined by the DSA paired with the CTIs. In the next prototype release, at M34, the CAUTHENG component will be further extended to be part of the Service Usage Control Adapter, to protect the usage of the C3ISP analytics services as well.

5.7.2. Maturation status

The original architecture and functionality of the CAUTHENG component were described in detail in deliverable D8.1, and they have been extended at M24 to deal with the features of the C3ISP scenario, and to meet the Requirements initially defined in deliverable D7.1 and refined for the CAUTHENG component in deliverable D8.1.

In the following we describe the maturations that have been addressed at M24, and we list the maturations that are still to be addressed to allow the full exploitation of the CAUTHENG in the C3ISP framework.

5.7.2.1. *Access Requests with Multiple Resources*

The first maturation of the CAUTHENG component required to accommodate the C3ISP scenario concerns the content of access requests. As a matter of fact, in the original CAUTHENG component, an access request involves the control on the usage of a **SINGLE** resource. Instead, an access request in C3ISP involve a **SET** of resources, the CTIs, which will be involved in the computation of an analytics to produce the required result. This set of CTI is passed by the user requesting the execution of the analytics either by explicitly indicating the ID of each or them in the request, or by defining the search criteria which will be used to query the DPOS to obtain such ID list. Instead, in other words, a usage session in C3ISP concerns multiple resources, while the original CAUTHENG component was designed to support session concerning a single resource only.

Hence, the original architecture of the CAUTHENG component has been updated in order to:

- Accept access requests including list of resource IDs instead of a single ID;
- Perform the policy evaluation process for each of the resource in such resource list, both at request time and continuously while the usage of such CTIs is in progress;
- Keep track of which resources are involved in each usage session;
- Combine the access decision results produced for the set of CTIs involved in a single request in order to obtain the final set of CTIs that can be exploited to compute the analytics while respecting the usage control policies of all the CTIs in the request.

5.7.2.2. *Usage Control Policy Refinement*

The second maturation required to deal with the C3ISP scenario concerns the kind of constraints that can be defined in the usage control policies paired with the data, and the related policy evaluation process. As a matter of fact, the original CAUTHENG component, manages access requests involving a single resource only and, consequently, is able to evaluate policies defining constraints that take into account the attributes related to this single resource, to the access requestor and to the environment. Instead, the C3ISP scenario requires that the usage control policies are able to express constraints concerning not only the resources they are paired to, but also third resources. In particular, the policy paired with a specific CTI C includes also rules which restricts the set of other CTIs with which C can be processes. In other words, the owner of a CTI could define policy rules to avoid that his CTI is used to perform an analytic with other specific CTIs, for instance produced by a competitor.

Hence, the original architecture of the CAUTHENG component has been updated in order to evaluate the new rules of the policy, and to properly combine the set of results related to the CTIs included in the same access request to determine the set of CTIs that will be used to perform the analytics.

5.7.2.3. *Requirement C3ISP-ComCAE-004 : Exploit Contextual Information*

The third maturation to be performed on the CAUTHENG component is determine by Requirement C3ISP-ComCAE-004 defined in deliverable D8.1, which was only partially met by the original CAUTHENG component. In particular, this requirement states that the component should be able to retrieve the contextual information for the evaluation of the DSA paired with the CTIs. This requirement was marked as partially met because the original CAUTHENG component can be configured to exploit external source of information called Attribute Managers, for retrieving the current values of the attributes which represents the context in which the access and the usage of the CTIs is being performed. This integration only requires the development of proper additional components, called Policy Information

Points, which can be seen as plugins allowing the CAUTHENG component to interact with Attribute Managers.

At M24 we identified two main sources of contextual information that can be used for the usage control policies:

- The Identity Service, which is paired with a Lightweight Directory Access Protocol (LDAP) service, providing information about the users of the C3ISP platform
- The MYSQL Service, which is a new service that has been deployed in the C3ISP platform to manage the some of the attributes concerning the resources.

Hence, the original architecture of the CAUTHENG component has been updated in order to add two PIP which allow the interaction with the AMs previously described.

5.7.2.4. Requirement C3ISP-ComCAE-010: Exploit User ID

The fourth maturation of the CAUTHENG component that has been implemented is related to Requirement C3ISP-ComCAE-010 defined in deliverable D8.1, which was only partially met by the original CAUTHENG component. This requirement states that the CAUTHENG component must be able to exploit the user ID in the usage decision process. As a matter of fact, the user who exploits the C3ISP platform is authenticated when he submits his access request, and the related user ID is passed to the CAUTHENG component by representing it as an attribute of the user. This ID is then exploited to retrieve further attributes concerning the user by querying the Identity service.

5.7.2.5. Other Maturations still to be addressed

Other maturations of the CAUTHENG component are required to address all the features of the C3ISP scenario and to meet all the Requirements defined in deliverable D8.1 for the CAUTHENG component. These maturations will be addressed in the next release of the prototype, at M34, and concerns the fulfilment of Requirements C3ISP-ComCAE-07, C3ISP-ComCAE-11, and C3ISP-ComCAE-12.

5.7.3. Requirement Analysis at M24

Table 6 – Continuous Authorization Engine Tool Requirements Status

ID	MET	Description
C3ISP-Com-CAE-001	YES	The current version of the tool is able to enforce both the access and usage control policies embedded in the DSA paired with the data shared with the Prosumers which define the controls to be performed to regulate the data sharing
C3ISP-Com-CAE-002	YES	The current version of the tool is able to evaluate the DSA paired with the data, to perform the decision process and determine whether to grant the access or not.

C3ISP-Com-CAE-003	YES	The current version of the tool is able to allow usage control on the shared data by continuously evaluating the DSA paired with the data, and by performing the decision process when the access context is changed in order to determine whether to interrupt the access in progress or not.
C3ISP-Com-CAE-004	YES	The tool has been configured in order to retrieve the contextual information from the Identity Manager Service and from the MYSQL service, and to process them in order to be exploited for the evaluation of the DSA paired with the data.
C3ISP-Com-CAE-005	YES	The tool is able to evaluate the DSA to support the Obligation Engine to decide whether an obligation expressed in the DSA (e.g., sending a notification) must be performed before, during, or after the end of the operation into.
C3ISP-Com-CAE-006	YES	The tool is able to evaluate the DSA to support the Obligation Engine to decide whether an obligation expressed in the DSA (which can be used to express data manipulation operations) must be performed before, during, or after the end of the operation.
C3ISP-Com-CAE-007	PARTIALLY	A further component which compute the risk of data sharing must be developed. This component should act as AM. The tool must be configured to consider the risk of data sharing as a new attribute.
C3ISP-Com-CAE-008	YES	The language currently supported by the tool is UPOL, an extension of XACML developed within the Coco Cloud EU FP7 project.
C3ISP-Com-CAE-009	YES	The tool is able to evaluate the DSA to support the Obligation Engine to support the Obligation Manager to enforce the obligations expressed in the DSA, which can be used to express data manipulation operations on the result of the operations.
C3ISP-Com-CAE-010	YES	The tool has been configured in order to exploit the user ID obtained in the authentication process in the access and usage decision process by representing it as an attribute of the user, and by using it to retrieve further attributes of the user.
C3ISP-Com-CAE-011	PARTIALLY	The tool must be instrumented to support multi-tenancy.
C3ISP-Com-	PARTIALLY	The impact of the policy enforcement on the performance of the analytics operations depends on several factors.

CAE-012		The performance will be analysed for each use case and proper optimization will be proposed where necessary.
---------	--	--

5.7.4. First release of the component

The original architecture of the CAUTHENG component, which is extensions the XACML reference architecture described in [XACML] and has been described in D8.1, has been further extended in order to implement the maturations listed in the previous section required for addressing the requirement of the C3ISP platform and pilots.

In Figure 38, it shows the new internal architecture of the CAUTHENG component, and in the following we give a description of the new components as well as we report a short description of the original ones.

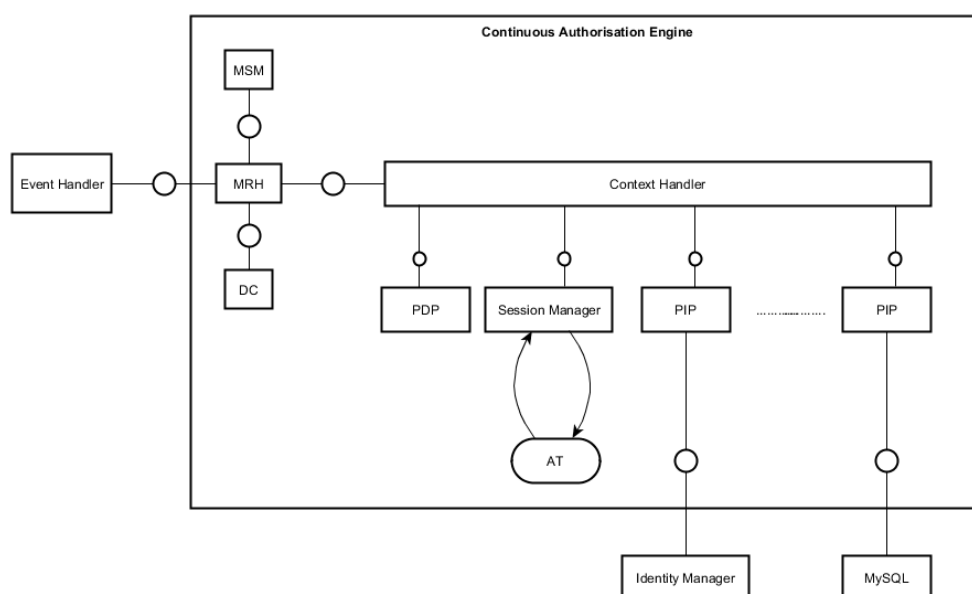


Figure 38: Components of the Continuous Authorization Engine

5.7.4.1. Multi-Resources Handler (MRH):

This component has been added in the CAUTHENG component architecture with respect of the version described in deliverable D8.1 as result of the maturation process, to address the specific features of the C3ISP platform and of the pilots use cases. As a matter of fact, the original CAUTHENG component was able to deal with access requests concerning one resource only, in order to authorize an operation to be performed on that resource taking into account in the access decision process the attributes of that resource only. Instead, in the C3ISP scenario, the execution of a single analytic function concerns a set of CTIs, not a single CTI at a time only, because one of the strengths of such analytic is to provide better results exploiting multiple data provided by distinct data sources.

Consequently, each access request sent to the CAUTHENG component concerns multiple resources at the same time, i.e., the set of CTIs on which the user wants to execute the specified analytics. Hence, with respect to the original version of the CAUTHENG component, the integration within the DSA Adapter component of the ISI subsystem required a modification of the architecture, i.e., the introduction of this new component called Multi-Resources Handler to deal with access requests involving multiple resources.

The MRH is hence a new interface of the CAUTHENG component, and it exposes new APIs accepting multiple resources access requests which are invoked by the Event Handler to perform the usage decision process on set of CTIs. However, the protocol is still defined by a subset of the usage control actions: *tryaccess*, *permitaccess*, *denyaccess*, *revokeaccess*, and *endaccess* (see [5, 6, 7] for further details), as for the CH component. The details concerning the interactions among the Event Handler and the MRH are described in deliverable D7.3, in Section 5.1.3.

As shown in Figure 39, the MRH intermediates the interactions between the Event Handler and the original CH. The MRH component is also in charge of extending the workflow of the policy evaluation process to accommodate the evaluation of access requests involving multiple CTIs. In particular, this component interacts with the Event Handler accepting access requests involving multiple CTIs, performs a loop cycle to ask to the original Usage Control systems to evaluate the usage control policy of each of them (i.e., interacting with the original CH exploiting the original protocol), collects all the decision responses, and interacts with the Decision Combiner component to combine such responses in order to define the final set of CTIs for the analytics to be performed. As shown in the following, the Decision Combiner component will determine the final set of CTIs according to given criteria.

5.7.4.1.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.1.2. Source Code Description

The MRH source code structure is depicted in Figure 39.

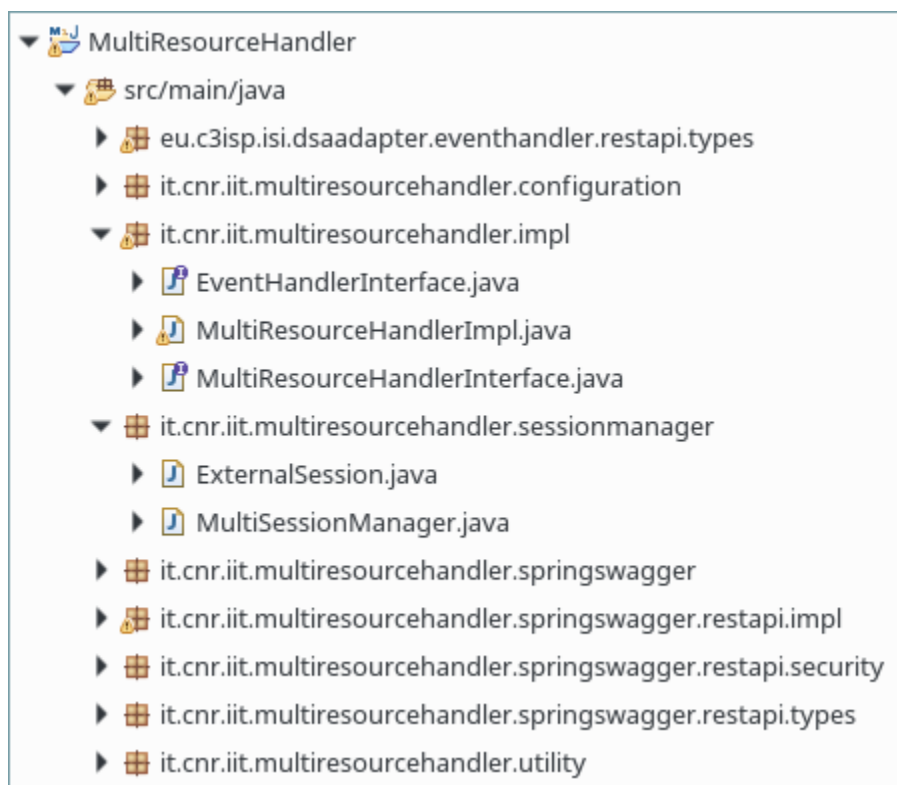


Figure 39: MRH code structure

The MRH exposes the set of APIs whose signature is reported in the `EventHandlerInterface.java` file, of the `it.cnr.iit.multiresourcehandler.impl` package, whilst

their actual implementation is reported in the `MultiResourceHandlerImpl.java` file of the same package. The `MultiSession Manager` code can be found in the related folder.

The MRH is interfaced with the Event Handler, according to a Publish/Subscribe schema. Hence at boot the MRH registers to the Event Handler via the subscribe API for a set of events, exposing to the Event Handler the REST API to be called whenever a processable message has been received (`rest_callback`), together with the type of event (`event_type`). The `it.cnr.iit.multiresourcehandler.utility` package includes a set of utility functions, such as the message tracker, which keeps track of the message exchanged with the Event Handler.

5.7.4.2. *Decision Combiner (DC)*

The Decision Combiner component has been added in the CAUTHENG component architecture with respect to the original version described in deliverable D8.1 as a result of the maturation process. As a matter of fact, this component has been added to provide a functionality to manage multi resource access requests.

The DC is invoked by the MRH after that the decision processes concerning each single CTI have been made, in order to determine the final set of CTIs on which the requested analytics will be performed that will be returned to the Event Handler. In particular, as explained in Section 5.7.2.2, the CTI protection needed for the C3ISP pilots states that the DSA paired with each CTI *C* includes rules which define constraints concerning the set of CTIs to which *C* can belong to in order to execute a given analytics. In other words, such policy could state that *C* cannot be processed by a given analytics in a set of CTIs *S* where the other member of *S* does not satisfy a set of rules, still defined on the basis of the attributes paired with such CTIs. For instance, a CTI producer could add to the DSA paired to his CTIs a rule to avoid that such CTIs are used to perform any analytic involving the CTIs produced by a competitor.

The main aim of the DC component is to determine the set of CTIs on which a given analytics will be executed in such a way that a specified objective function is satisfied while the policies paired with all the CTIs included in such set are all satisfied. In general, many distinct objective functions can be defined, depending on the requirement of the specific analytics to be performed or on the preferences of the user who requested to perform such analytics. A simple example of objective function could be the maximization of the number of CTIs involved in the analytics. Another objective function that could be defined could be aimed at determining the larger set of CTIs that have not been anonymized. In the prototype released at M24 a very simple objective function is provided. More complex objective functions that will be defined according to Pilots requirements, will be available in the second release of the prototype.

5.7.4.2.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.2.2. Source Code Description

The decision combiner is currently under development and will be described in the following deliverables.

5.7.4.3. *Multi Session Manager (MSM)*

The Multi Session Manager component is the third component which has been added to the CAUTHENG component architecture with respect to the original version described in deliverable D8.1 as result of the maturation process. As for the previous two components, the MSM has been added to provide functionality to manage multi resource access requests.

The main task of this component is to keep track of a set of data to connect the usage session of each single CTI (which is managed by the original SM) with the multi resource access request it belongs to.

5.7.4.3.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.3.2. Source Code Description

The Session Manager is a subcomponent of the MRH, hence its code is part of the MRH project, depicted in Figure 39. The code of the MultiSession handler is reported in the MultiSessionManager.java class.

5.7.4.4. Context Handler (CH)

The Context Handler was present in the CAUTHENG architecture from its original version. In the original version, the CH was the entry point of the CAUTHENG component, managing the protocol for communicating with the Event Handler. This protocol is defined by a subset of the usage control actions: *tryaccess*, *permitaccess*, *denyaccess*, *revokeaccess*, and *endaccess* (see [5, 6, 7] for further details). In the current version of the CAUTHENG component, the CH communicates with the MRH, which intermediates the interactions with the Event Handler. As explained before, this change was required to enable the CAUTHENG component to deal with multi resources access requests. The CH coordinates the interactions among the original internal components of the CAUTHENG architecture for the execution of the policy evaluation process, while the MRH is in charge of coordinating the interactions among the CH and the new components of the CAUTHENG architecture;

5.7.4.4.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.4.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.1.3. The CH code is defined in the `iit.cnr.it.usagecontrolframework.contexthandler` package, reported in Figure 40.

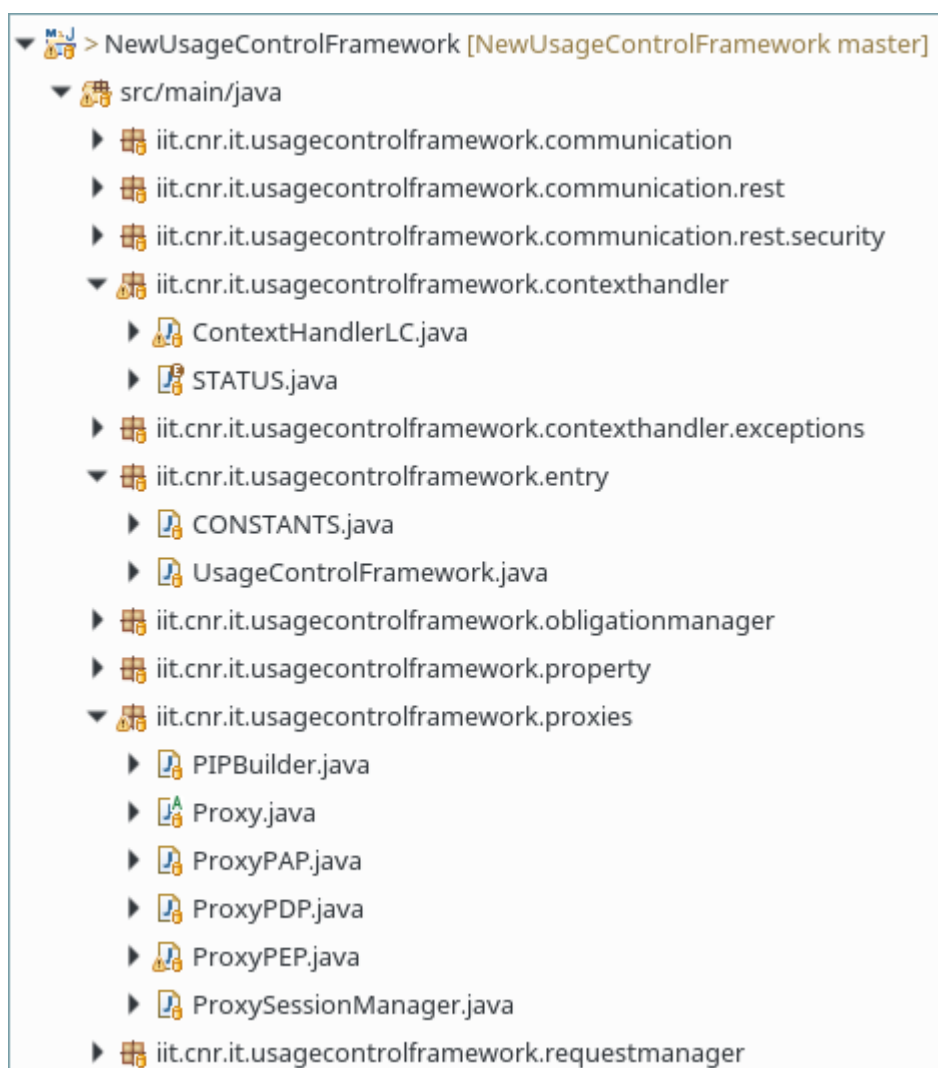


Figure 40: Core of the CAUTHENG source code

Despite its complexity, the CH is implemented with clear understanding through a single Java file.

5.7.4.5. Session Manager (SM)

The Session Manager was present in the CAUTHENG architecture from its original version. The task assigned to this component and its behaviour have not been changed with respect to the original version of the component. As a matter of fact, this component is still responsible for keeping track of the ongoing usage sessions, i.e., of the access that are currently in progress, and it exploits an Access Table (AT) to store the meta-data regarding these sessions. It is the key component of the continuous authorization phase, and it represents an extension with respect to the XACML reference architecture defined in [XACML];

5.7.4.5.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.5.2. Source Code Description

The code of the session manager is implemented as a set of interfaces exploiting the proxy design pattern, to ensure the communication with different physical configurations of the

database implementing the SM. The source code can be found in the ProxySessionManager.java class file, shown in Figure 40.

5.7.4.6. Policy Decision Point (PDP)

The Policy Decision Point was present in the CAUTHENG architecture from its original version. The task assigned to this component has not been changed with respect to the original version of the component. The PDP evaluates security policies and produces the access decision. The security policies are expressed using the XACML standard because the usage control specific features are managed by the CH and by the SM;

5.7.4.6.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.6.2. Source Code Description

For the implementation of the Policy Decision Point we adopted an existing tool, WSO2 balana¹⁰. This tool provides all the features of a XACML engine described by the XACML standard [XACML] Since the CAUTHENG implementation follows the XACML standard, other tools could be easily integrated in the CAUTHENG component replacing WSO2 balana. The interface to the Balana PDP is also implemented through the proxy design pattern. The implementation can be found in the class ProxyPDP.java reported in Figure 40.

5.7.4.7. Attribute Managers (AMs)

Attribute Managers are components which manage the attributes that are required to evaluate the usage control policy, allowing to retrieve their current values, and sometimes to update them. Each application scenario has its own attributes, which describe the security relevant features of subject, resource and environment for that specific context. Consequently, each scenario identifies a potentially distinct set of Attribute Managers.

The maturation process of the CAUTHENG component identified two relevant Attribute Managers providing attributes that are exploited by Usage Control policies defined in the DSAs of the C3ISP pilots, namely:

- Identity Manager Service: this component is provided by the Common Security Services subsystem, and it provides information about the C3ISP users through an LDAP service, such as the organization the user works for, the projects the user is involved in, and others.
- MYSQL service: this component has been added to the DSA Adapter component architecture with respect to the original version described in deliverable D8.2.

The two previous AMs have their own protocols, and we defined two specific Policy Information Points for interacting with them exploiting those protocols;

5.7.4.7.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

¹⁰ WSO2 Balana implementation: <https://github.com/wso2/balana>

5.7.4.7.2. Source Code Description

The MYSQL Attribute Manager, instead, is based on a standard MySQL database. The offered APIs are the standard JDBC functions to query the DB via SQL queries.

5.7.4.8. Policy Information Points (PIPs)

The Policy Information Points are the interfaces exploited by the CH for interacting with the Attribute Managers to retrieve the current values of the attributes required to evaluate the Usage Control Policy. PIPs are required because distinct Attribute Managers typically support different protocols, and PIPs mimic a plug-in architecture to let the CH be unaware of that specific protocols. In particular, the proposed architecture includes a set (chain) of PIPs which provide the same interface to the CH (*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.

Since, the maturation process of the CAUTHENG component identified two relevant Attribute Managers, described in the previous section, the two related PIPs have been developed:

- **LDAP PIP:** This PIP is the dual component of the LDAP AM. It retrieves information related to the user accessing the C3ISP services, such as name, role, department, etc.
- **MySQL PIP:** This PIP is the dual component of the MySQL AM. It is used to query different kind of information stored in a DB, whose parameters and structure are set in a static configuration file. Since the MySQL AM does not include any mechanism for subscription, the PIP needs to periodically query the value of the attributes that are mutable. The period is a parameter that can be set.

5.7.4.8.1. Link to Source Code

The link to SVN for the source code is available at: <https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/CAUTHENG>

5.7.4.8.2. Source Code Description

The source code of the two PIPs are two separated project whose structure is depicted respectively in Figure 41 and Figure 42. This choice is due to the fact that PIPs are implementation of an abstract class and are specific for every use case, needing ad hoc development.

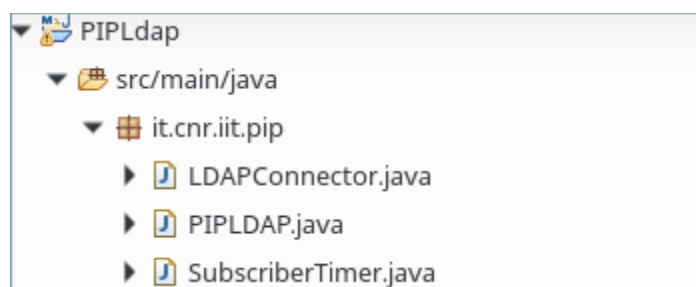


Figure 41: PIP LDAP code structure



Figure 42: PIP MySQL code structure

5.8. *Obligation Engine*

5.8.1. Component description

The Obligation Engine is a component that allows the execution of pre-determined operations when specific conditions occur. Such operations are defined as *Usage Control Obligations* that are included in an enforceable policy, that is, the DSA associated to a specific piece of information. The enforceable policy is expressed in UPOL language, a security policy language developed in the Coco Cloud EU FP7 project (GA #610853) and under extension in C3ISP.

5.8.2. Maturation status

The Obligation Engine implementation has significantly matured has foreseen in the previous D8.1. The main driver such process is the need to cope with data volume and velocity, typical characteristics of big data deployments.

We recall that the obligation can be described as follows:

- Usage Control Obligation = do *Action* when *Trigger*

Where *Trigger* is defined by:

- *Trigger* = *Event* AND *Condition*

For this reason, the actual implementation relies on two different engines, the *trigger* and the *action engines*. The trigger engine is the entry point of the obligation engine, as it caters for a method to submit a new obligation. Once called, this method sets up a new trigger set, configured with the associated action(s) according to the obligation definition.

The trigger and action engines use a library, MapDB¹¹, that provides concurrent Maps, Sets and Queues backed by disk storage or off-heap-memory. It is a fast and easy to use embedded Java database engine. MapDB allows for handling dynamic registration of actions and management/set up of triggers.

Notably, following to design review sessions held during Y2, it was possible to analyse and decompose the Obligation Engine use cases and observe strong similarities among the Obligation Engine, the DMO Engine and the Enterprise pilot C3ISP Gateway. For this reason, the implementation of these 3 components share some parts, to allow for a rational usage and allocation of development resources.

¹¹ <https://github.com/jankotek/mapd>

5.8.3. Requirement Analysis at M24

The following Table 7 analyses the fulfilment of Obligation Engine requirements as defined in deliverable D8.1. A general improvement on the fulfilment of practically all requirements can be observed, going towards the achievement of all objectives for the component by the end of the project.

Table 7: Obligation Engine requirement analysis.

ID	MET	Description
C3ISP-Com-OBE-001	YES/ONGOING	The current implementation supports the triggers and actions needed at M24 for data sharing operations. More triggers are expected to be defined and implemented by the end of the project.
C3ISP-Com-OBE-002	YES/ONGOING	The current implementation supports the triggers and actions needed at M24 for data analytics operations. More triggers are expected to be defined and implemented by the end of the project.
C3ISP-Com-OBE-003	PARTIALLY	The Anonymization Engine results fully integrated with the Obligation Engine while more work is needed for the FHE component.
C3ISP-Com-OBE-004	YES (unchanged)	Certain audit obligations (generation of audit trails, email sending etc.) are already supported by the current implementation. Adaptations or further extensions may be needed, though.
C3ISP-Com-OBE-005	YES/ONGOING	The actual connection with the DMO Engine exists but it will be enhanced when the DMO Engine will be fully implemented.
C3ISP-Com-OBE-006	YES/ONGOING	The Obligation Engine can trigger any necessary operation needed to estimate the risk associated to an operation, when this process is modelled as a supported obligation. Concrete measures are yet to be identified and thus implemented.
C3ISP-Com-OBE-007	PARTIALLY	The actual connection with the DMO Engine exists but it will be enhanced when the DMO Engine will be fully implemented.
C3ISP-Com-OBE-008	YES/ONGOING	At this stage, the actual implementation of the Obligation Engine seems to cope with the required performance requirements. Further analysis will be conducted before the end of the project.

5.8.4. First release of the component

The first release of the Obligation Engine is composed of two different modules, the Trigger and the Action Engines. Both run as Java applications powered by the Spring framework, exposing RESTful interfaces.

The source code of the two project is structured similarly. Both rely on the MapDB library to maintain and persist configuration information. They expose methods for the dynamic management of triggers and actions. The respective RESTful methods are detailed in the following:

Table 8: Trigger Engine RESTful methods.

Method Name	Note	Parameter Example
<i>/v1/obligation</i>	A POST request to the URL with the parameters as for the example will register a new obligation in the Trigger and Action engine. The return element of the call is a HTTP status code, with 200 meaning a successful registration.	HTTP Form with: - message: { "Attribute" : [{ "parameterId" : <parameter name>, "parameterValue" : <value>, "DataType" : "string" }], "triggerId" : <value>, }
<i>/v1/trigger</i>	A POST request to the URL with the parameters as for the example will register a new trigger implementation. The input JSON contains the endpoint to call the trigger, an triggerId identifier and all required parameters for the call. The return element of the call is a HTTP status code, with 200 meaning a successful registration.	HTTP Form with: - message: { "Attribute" : [{ "parameterId" : <parameter name>, "parameterPassingMode" : <GET PATH QUERY>, "DataType" : "string" }], "triggerId" : <value>, "endPoint" : <value>, "httpMethod" : <HTTP VERB> }

<i>/v1/trigger</i>	A GET request to this endpoint, will return a list of actions registered in the Trigger Engine.	No parameters are needed.
<i>/v1/activate</i>	A POST request to this endpoint (once user is authenticated), if authorised, will set up a trigger (specified with triggerId) with the specified parameters using the metadata format as specified in the next cell.	HTTP Form with: - message: { "Attribute" : [{ "parameterId" : <parameter name>, "parameterValue" : <value>, "DataType" : "string" }], "triggerId" : <value>, }
<i>/v1/trigger/trigger-at-time</i>	A POST request to this endpoint will set up a new trigger at a specified time.	HTTP query string: dpo_id : <dpo_id value>
<i>/v1/eventNotification</i>	This method allows to receive asynchronous notifications from the Event Handler.	HTTP Body: { "additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "sessionId" : "string" }

Table 9: Action Engine RESTful methods.

Method Name	Note	Parameter Example
<i>/v1/action</i>	A POST request to the URL with the parameters as for the example will register a new action. The input JSON contains the endpoint to call the action, an actionId identifier and all required parameters for the call. The return	HTTP Form with: - message: { "Attribute" : [{ "parameterId" : <parameter

	element of the call is a HTTP status code, with 200 meaning a successful registration.	<pre> name>, "parameterPassingMode" : <GET PATH QUERY>, "DataType" : "string" }], "actionId" : <value>, "endPoint" : <value>, "httpMethod" : <HTTP VERB> } </pre>
<i>/v1/action</i>	A GET request to this endpoint, will return a list of actions registered in the Action Engine.	No parameters are needed.
<i>/v1/execute</i>	A POST request to this endpoint (once user is authenticated), if authorised, will execute an action (specified with actionId) with the specified parameters using the metadata format as specified in the next cell.	<p>HTTP Form with:</p> <ul style="list-style-type: none"> - message: <pre> { "Attribute" : [{ "parameterId" : <parameter name>, "parameterValue" : <value>, "DataType" : "string" }], "actionId" : <value>, } </pre>
<i>/v1/action/delete-dpo</i>	A DELETE request to this endpoint will trigger the deletion of a DPO from the ISI.	<p>HTTP query string:</p> <pre> dpo_id : <dpo_id value> </pre>
<i>/v1/eventNotification</i>	This method allows to receive asynchronous notifications from the Event Handler.	<p>HTTP Body:</p> <pre> { "additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "sessionId" : "string" } </pre>

5.8.4.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/obligation-engine/trigger-engine>
(binary)

5.8.4.2. Source Code Description

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.1.4.

The structures of the Trigger and Action engines are similar. They both share the same persistency management (based on MapDB, as mentioned) and the RESTful method structure.

More in details:

The main packages *eu.c3isp.oe.te* and *eu.c3isp.oe.ae* contain the *ApplicationDeployer* classes, in charge of enabling the Spring framework activation for the web applications.

Packages *eu.c3isp.oe.te.persistency* and *eu.c3isp.oe.ae.persistency* contain the persistency management implementation using MapDB.

The packages *eu.c3isp.oe.te.restapi.impl* and *eu.c3isp.oe.ae.restapi.impl* contain the definition and implementation of the Event Listener class, required for interacting with the Event Handler, together with all management classes for, respectively, triggers and actions known by the engines.

Package *eu.c3isp.oe.te.restapi.types* and *eu.c3isp.oe.ae.restapi.types* cater for all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

Package *eu.c3isp.oe.te.restapi.triggers* contains all trigger implementations available in the trigger engine. Similarly, *eu.c3isp.oe.ae.restapi.actions* hosts the build-in action implementations. Additional triggers and actions may be registered, by means of the dynamic trigger/action management.

Lastly, *eu.c3isp.oe.te.restapi.security* and *eu.c3isp.oe.ae.restapi.security* have the Spring security initialization.

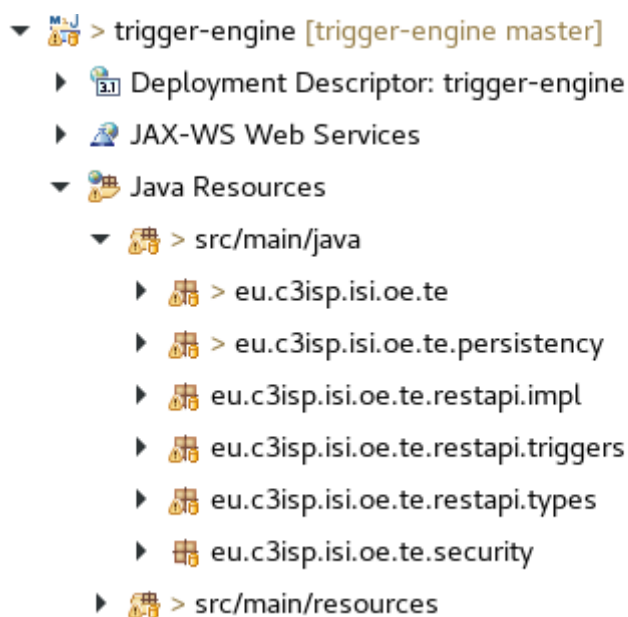


Figure 43: Trigger Engine (part of the Obligation Engine) source code packages from the Eclipse IDE.

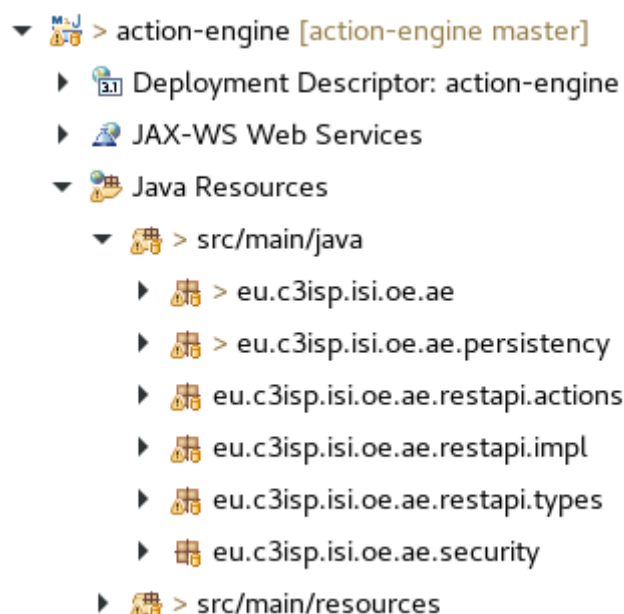


Figure 44: Action Engine (part of the Obligation Engine) source code packages from the Eclipse IDE.

5.9. *Data Manipulation Operation Engine*

5.9.1. **Component description**

The Data Manipulation Operation (DMO) Engine is a component in charge of executing the Data Manipulation Operation as prescribed by the DSA. In fact, besides determining whether the data can be accessed or not by the requestor, the decision process also determines a set of operations that must be executed on such data before being released to the requestor.

Following to design review sessions held during Y2, it was observed a similarity in planned use cases among the DMO Engine, the Obligation Engine (and more specifically, the Action Engine) and the Enterprise pilot C3ISP Gateway. For this reason, the implementation of these three components share some parts, to allow for a rational usage and allocation of development resources.

The DMO Engine is implemented using the Spring framework family. It relies on the latter for exposing a REST interface, as well as for configuration management and dependency injection.

5.9.2. **Maturation status**

The DMO Engine is implemented and available at M24. It is fully integrated with the Anonymization Toolbox and only a proof-of-concept (not in production) integration is available for the Homomorphic Encryption Engine.

The DMO Engine uses a library, MapDB¹², that provides concurrent Maps, Sets and Queues backed by disk storage or off-heap-memory. It is a fast and easy to use embedded Java database engine. MapDB allows for handling dynamic registration of actions and management/set up of triggers.

¹² <https://github.com/jankotek/mapd>

5.9.3. Requirement Analysis at M24

The DMO Engine was not part of components analysed in D8.1, therefore this section will contain a list of requirements for the component to be fulfilled by the end of the project.

ID	Priority	Requirement
C3ISP-Com-DMOE-001	MUST	The DMO Engine must be able to call all required data manipulation operations.
C3ISP-Com-DMOE-002	MAY	DMO Engine must be callable by the ISI API components.

The requirement analysis may be detailed as follows:

ID	MET	Description
C3ISP-Com-DMOE-001	PARTIALLY	The integration with the Homomorphic Encryption solution is not complete and will be achieved by M25, the integration with Anonymization Toolbox is complete at this stage.
C3ISP-Com-DMOE-002	YES	The DMO Engine is integrated with the Event Handler thus it is enabled to invoke DMOs as required.

5.9.4. First release of the component

The first release of the DMO Engine runs as a Java application powered by the Spring framework. Its implementation relies on two main group of classes. One, the Event Listener, communicates with the Event Handler, sending and receiving messages towards and from the other DSA Adapter components. The other group is in charge of the implementation of the functionalities exposed by the DMO Engine. Its RESTful API methods are implemented by means of a number of classes, as detailed in the following Table 10.

Table 10: DMO Engine RESTful methods.

Method Name	Note	Parameter Example
<i>/v1/action</i>	A POST request to the URL with the parameters as for the example will register a new DMO operation. The input JSON contains the endpoint to call the action, an actionId identifier and all required	HTTP Form with: - message: { "Attribute" : [{ "parameterId" : <parameter

	parameters for the call. The return element of the call is a HTTP status code, with 200 meaning a successful registration.	<pre> name>, "parameterPassingMode" : <GET PATH QUERY>, "DataType" : "string" }], "actionId" : <value>, "endPoint" : <value>, "httpMethod" : <HTTP VERB> } </pre>
<i>/v1/action</i>	A GET request to this endpoint, will return a list of actions registered in the Action Engine.	No parameters are needed.
<i>/v1/execute</i>	A POST request to this endpoint (once user is authenticated), if authorised, will execute an action (specified with actionId) with the specified parameters using the metadata format as specified in the next cell.	<p>HTTP Form with:</p> <ul style="list-style-type: none"> - message: <pre> { "Attribute" : [{ "parameterId" : <parameter name>, "parameterValue" : <value>, "DataType" : "string" }], "actionId" : <value>, } </pre>
<i>/v1/action/anonymize-dpo</i>	A POST request to this endpoint will trigger the anonymization of a DPO from the ISI.	<p>HTTP Form with:</p> <ul style="list-style-type: none"> - message: <pre> { "Attribute" : [{ "parameterId" : <parameter name>, "parameterValue" : <value>, "DataType" : "string" }], "MechanismName" : <value>, "PrivacyParameter" : <value> } </pre>

<i>/v1/eventNotification</i>	This method allows to receive asynchronous notifications from the Event Handler.	<p>HTTP Body:</p> <pre>{ "additionalProperties" : { "Attribute" : <any desired value>, ...}, "eventType" : <event type value>, "sessionId" : "string" } }</pre>
------------------------------	--	---

5.9.4.1. *Link to Source Code*

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/dmo-engine/dmo-engine> (binaries)

5.9.4.2. *Source Code Description*

The list of the APIs and their description is reported in deliverable D7.3, in Section 5.1.5.

The source code is depicted in Figure 45.

More in details:

The main packages *eu.c3isp.isi.dmo* contains the *ApplicationDeployer* classes, in charge of enabling the Spring framework activation for the web application.

The persistency management (based on MapDB, as mentioned) is achieved by means of classes in package *eu.c3isp.isi.dmo.persistency*.

The packages *eu.c3isp.isi.dmo.impl* contains the definition and implementation of the *Event Listener* class, required for interacting with the *Event Handler*, together with all management classes for the data management operations of the engine.

Package *eu.c3isp.isi.dmo.restapi.types* caters for all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

Package *eu.c3isp.isi.dmo.actions* contains all DMO adapters implementations needed to call the actual DMO implementations.

Lastly, *eu.c3isp.isi.dmo.security* has the Spring security initialization.

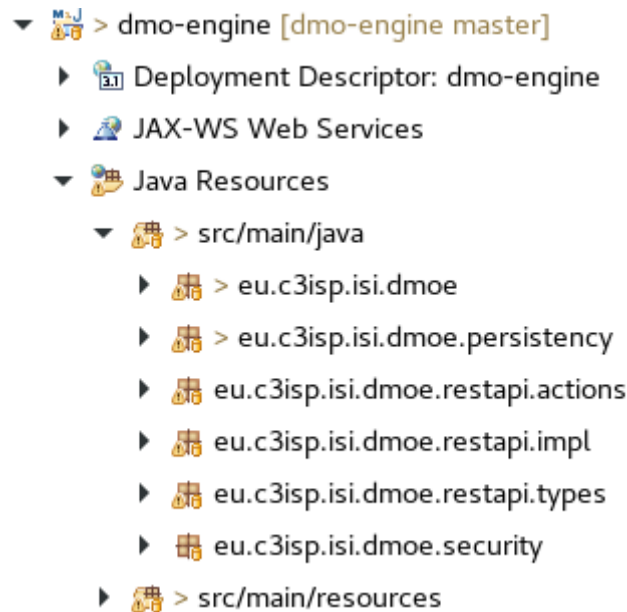


Figure 45: DMO Engine source code packages from the Eclipse IDE.

5.10. Bundle Manager

5.10.1. Component description

The **Bundle Manager (BM)** is a module of the *DSA Adapter*. It aims at posting, deleting and retrieving packets of data called *DPOs (Data Protected Object)*. Such packets consist of four files put in a ZIP file which is encrypted with AES and then sent to *DPOS (Data Protected Object Storage)*. These packets can later be retrieved from the *DPOS* component or be deleted.

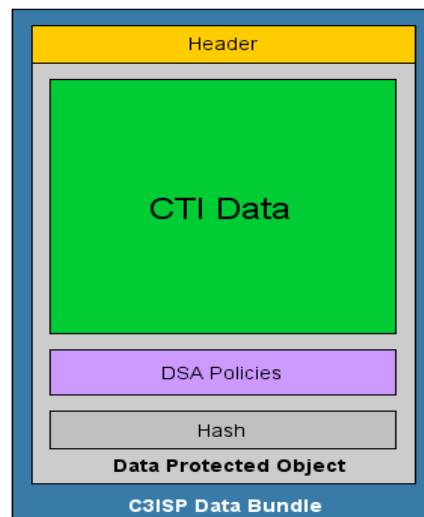


Figure 46: The structure of a bundle data

5.10.2. Maturation status

The main functions of the *BM* component have been implemented, almost finished and completed. The *BM* component consists of encrypting and packaging CTI data, metadata and

DSA data in only one packet before invoking DPO API to store them in DPOS platform. The encryption step is performed by the *Key & Encryption Manager* component (part of the CSS subsystem, see D7.3).

The complete version for BM component is indirectly interacted with IAI platform. Indirectly word in this case means that when BM receives a request for Kreyvium encryption, BM will invoke a request to K&E Manager to encrypt it. At this moment, the transciphering step might be executed in order to get ready and generate in advance the data encrypted to FHE form. These data will be stored in a specific folder for analysing with FHE later.

During CTI Bundle creation, the current version of the BM does not compute the hash with a cryptographic hash function which guarantees the data integrity (i.e. data have not been modified by unauthorized parties). It currently uses a hash function provided by Java which does not offer this functionality. SHA-2 cryptographic hash function should be employed in the next version.

5.10.3. First release of the component

The BM is integrated in one API accessible via <https://isic3isp.iit.cnr.it:8443/bundle-manager>

It permits five operations:

- Create a CTI Bundle for data storage (simple encryption)
 - Function: `/v1/bundle/create/{requestId}`
 - This operation allows creating a DPOs from a metadata file, a CTI data file and DSA file. According to requirement,
- DSA file is encrypted by a couple of AES keys which are identified by the unique DSA identifier,
- CTI file is encrypted by AES keys which can be used for IAI platform.
- Metadata file is in clear form for searching CTI purpose.
- Retrieve full CTI bundle
 - Function: `/v1/bundle/read/{dposId}/{payloadFormat}`
 - This operation consists of retrieving a complete bundle object (Metadata, CTI file, DSA file and hash file) which is identified by *dposId*. In function of the specification field *payloadFormat* defined in the path of request, this operation can give a bundle data in encrypted form (`EncryptedFormat`) or in clear form (`ClearFormat`).
- Delete a CTI bundle
 - Function: `/v1/bundle/delete/{dposId}`
 - This operation consists of deleting the bundle data. The function “undo” for deleting is not provided. So, this action is irreversible.
- Create a CTI Bundle for data manipulation (multiple encryption)
 - Function: `/v1/dmo/create/{requestId}/transciphering`
 - This operation consists of encrypting a specific data with Kreyvium algorithm (a symmetric encryption), which is contained in the CTI file before making transciphering to Homomorphic encryption form. That is an important step before performing this data with FHE technology. In this version, the specific data is IP in *dest* field in CTI file.
- Receive an event from Event Handler for processing CTI Bundle
 - Function: `/v1/notifyEvent`

This function consists of receiving all the requests from Event Handler for manipulating with CTI data.

Generic workflows are given and described in the deliverable D7.3, Section 4.1.6. The bundle manager is exposed at the following url:

<https://isic3isp.iit.cnr.it/bundle-manager/swagger-ui.html>

Bundle Manager API Model with Springboot REST API

CRUD Bundle API Model for CTI management

Created by Thanh-Hai NGUYEN
See more at <http://www-list.cea.fr/en/>
[Contact the developer](#)
[Apache 2.0](#)

bundle		Show/Hide	List Operations	Expand Operations
POST	/v1/bundle/create/{requestId}	create a bundle data with a required request Id, DSA Id, CTI data and CTI metadata		
DELETE	/v1/bundle/delete/{dposId}/	delete a CTI bundle with dposId		
GET	/v1/bundle/read/{dposId}/{payloadFormat}	get all data contained in bundle with a given DposID		
dmo		Show/Hide	List Operations	Expand Operations
POST	/v1/dmo/apply/{requestId}/transcribing	create a bundle data with a required request Id, DSA Id, CTI data and CTI metadata		
event-handler : Event Handler		Show/Hide	List Operations	Expand Operations
POST	/v1/notifyEvent	Subscribes a URL from notifications of Events with eventType		


[BASE URL: /bundle-manager , API VERSION: 1.0.0] 

Figure 47: Bundle Manager APIs

5.10.3.1. Link to Source Code

The code source is available at:

<https://devC3ISP.iit.cnr.it:8443/c3isp-wp8/isi/bundle-manager/bundle-manager-api.git>.

5.10.3.2. Source Code Description

The list of the APIs and the description for these API is reported in deliverable D7.3, in Section 5.1.6. The **Figure 48** show the structure of Bundle Manager code source.

The main package *fr.cea.bundle* contains the *ApplicationDeployer* class, in charge of enabling the Spring framework activation for the web application.

The package *fr.cea.bundle.client.restapi* contains the definition and implementation from client side for invoking the Bundle Manager APIs, this class is only used when receiving events from Event Handler component.

The package *fr.cea.bundle.components* caters for all the client interactions with others components via their APIs

The package *fr.cea.bundle.config* is used for API access configuration.

The package *fr.cea.bundle.helper* contains all the general methods and constants which are used in this Bundle Manager project

The package *fr.cea.bundle.models* contains all the model class definitions. Metadata object containers are defined here for each of the exposed REST methods.

The package *fr.cea.bundle.restapi* contains all the definition of API methods

The package *fr.cea.bundle.kecore.api* contains the interaction with K&E Manager API and same usage purpose for the package *fr.cea.bundle.kent.dpos.client* for DPOS component

Lastly, *fr.cea.bundle.security* and *fr.cea.bundle.sevices* has the Spring security initialization.

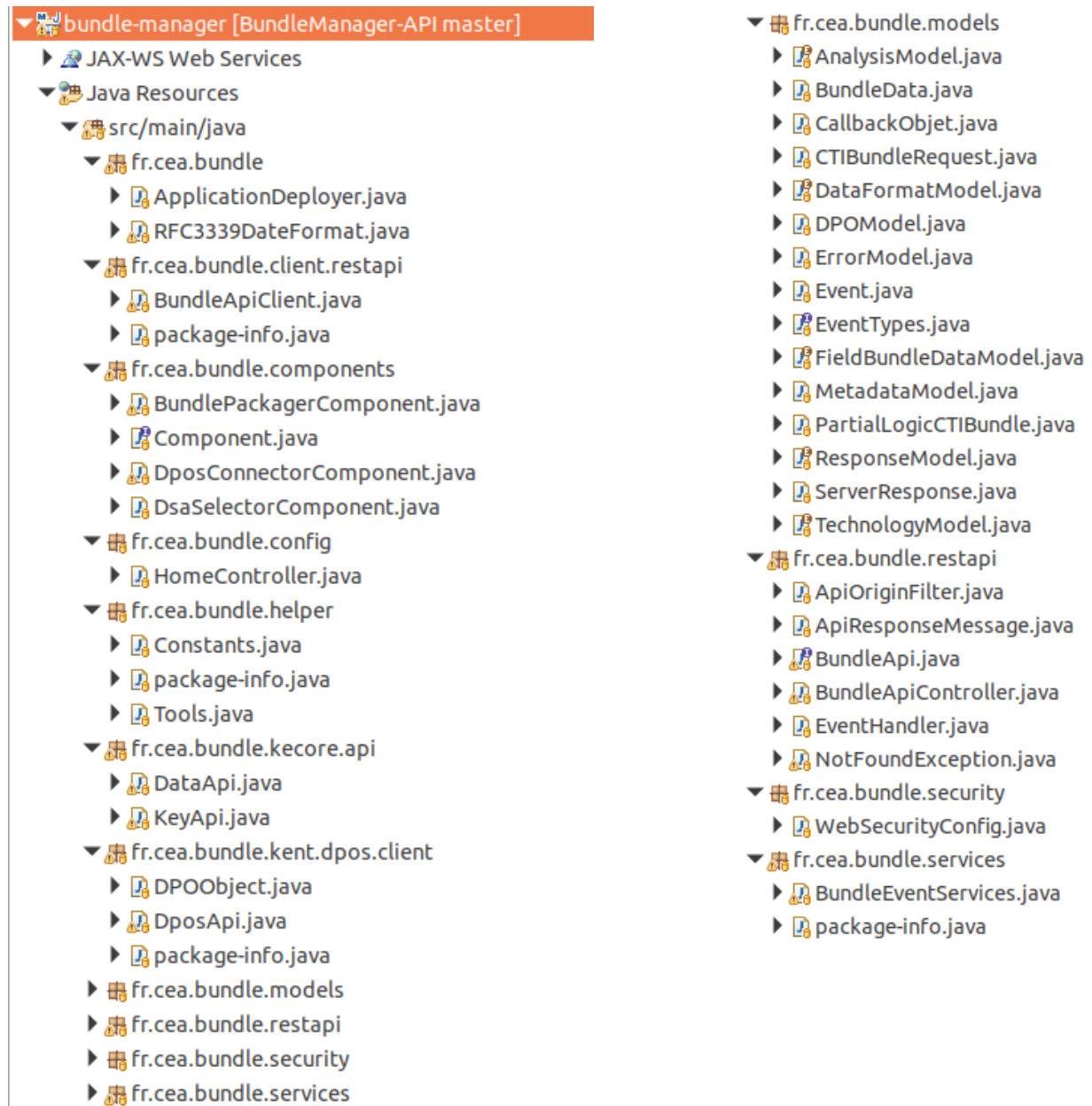


Figure 48: Bundle Manager code source structure

6. Collaborative Data Analytics: The Information Analytics Infrastructure (IAI)

The Information Analytics Infrastructure (IAI) allows Prosumers to request the execution of analytics services on the data protected and shared by the ISI. It supports both so called C3ISP-aware analytics services, jobs that can exploit the full capabilities of the C3ISP Framework, and so-called legacy analytics services (i.e. already existent analytics), that can run on the shared data but have limitations.

6.1. Specific Data Analytics Examples

This section describes the analytics that have been developed at M24. Further analytics will be developed for the second release of the prototype

6.1.1. Component description

6.1.1.1. *Monitoring of connections to malicious hosts.*

This analytic works on connection request logs and identifies whether the destination addresses belong to malicious hosts. Connection logs are directly taken from a router that use the Netflow V9 protocol and pushes the information to a client software that collects the logs.

This service is run with a combination of a DMO, more specifically, the homomorphic encryption one, which processes the data directly in the crypto-text keeping private the data analysed. See Section 8.2 for the complete status of this component.

6.1.1.2. *Monitoring of Domain Generation Algorithm DNS-request.*

This analytic works on DNS request logs and identifies whether domain names have been resolved within domains that refer to a Domain Generating Algorithm (DGA). These are used by malware to register new domains on the fly to avoid that malware depends on a fixed domain or an IP address that could be quickly blocked. Thus, the malware switches to a new domain at regular intervals and thus prevents that a new version of the malware is released.

6.1.1.3. *Detection of brute force/DDoS attacks.*

This analytic aims at detecting brute-force or DDoS attacks on the log of services, e.g., SSH. So, IPs found as malicious can be used by prosumers to block connections by setting up firewall policies rules.

6.1.1.4. *Spam Email analysis.*

This analytic takes as input a set of unsolicited email files in eml format, hence it extracts from these email files a set of numerical features, which are representative of the email structure, hence the emails are at first separated in campaigns, grouping the ones showing a similar structure, then they are separated in different classes on the base of the spammer goal.

6.1.2. Maturation status

6.1.2.1. *Monitoring of connections to malicious hosts.*

This analytic aims at detecting a given IP which is encrypted in FHE format belongs to the black list of IPs. the details are described in Section 8.2.2 of the same deliverable.

6.1.2.2. *Monitoring of Domain Generation Algorithm DNS-request.*

This analytic aims at detecting DNS requests used by malware to register new domains on the fly, e.g., www.fgd2iwya7vinfutj5wg5we.com. Compared to the first year of the project, this analytic has reached a good level of maturation that allows the analytic to properly process DNS-request logs belonging to the BIND¹³ open source software, which resolves DNS queries for users. In particular, the current maturation status for this analytic consists of:

- Managing request logs coming from BIND software that are inputted as CEF format;
- Using an internal script that processes each domain name resolved, which appears in the log, and identified if the resolved name is a DGA;
- Producing as out a list of DGA names found in the logs structured as CEF format keeping the original information of the log, for instance the timestamp when the requested has been done to the DNS.

6.1.2.3. *Spam Email Analysis*

The analytics aims at detecting, classifying and clustering spam emails according to the goal of the spammer and exploiting structural similarity. The maturation performed till M24 consists in the introduction of a mechanism based on deep learning for separating genuine emails from malicious one, allowing an accurate filtering HAM vs SPAM. Moreover, the analytics has been made compatible with data in STIX format, which is the common format currently used in C3ISP.

6.1.3. Requirement Analysis at M24

6.1.3.1. *Monitoring of Domain Generation Algorithm DNS-request.*

This analytic has met the requirement expressed in the D2.1 as ISP-UC-01 and ISP-US-01 and in particular, the need of the Monitoring of Domain Generation Algorithm DNS-request analytic for this pilot was expressed in the interview done to ISPs always described in D2.1.

6.1.3.2. *Detection of brute force/DDoS attacks.*

This analytic has met the requirement expressed in the D2.1 as ISP-UC-01 and ISP-US-01 and in particular, the need of the Detection of brute force/DDoS attacks analytic for this pilot was expressed in the interview done to ISPs always described in D2.1.

6.1.3.3. *Spam Email Analysis*

The analytics has met the requirements of D3.1 for CERT-US-05, showing effectiveness and accuracy in classifying spam emails according to the threat that they bring.

6.1.4. First release of the component

6.1.4.1. *Monitoring of Domain Generation Algorithm DNS-request.*

At m24 this analytic has been released and it is up and running within the IAI component of the C3ISP Framework. The current release of the component consists of two API, which are:

API	Description
/detectDGA	It takes domain-name logs and check if they are DGA using a third-

¹³<https://www.isc.org/downloads/bind/>

	party algorithm.
/matchDGA	It takes domain-name logs and check if they are DGA using a public source of known DGAs.

The analytic is exposed at the following url:

<https://iaic3isp.iit.cnr.it/monitoring-dga/swagger-ui.html#/dga45service45implementation>

In Figure 49, it is illustrated the swagger representation on this analytic showing the two APIs

The screenshot shows the Swagger UI for the 'DGA monitoring API'. The header is green with the 'swagger' logo and an 'Explore' button. Below the header, the API title 'DGA monitoring API' is displayed, followed by a description: 'API to discover DGAs from a public source site and ISP sources. Also, this API allows ISPs to share already known DGA or discovered.' The creator information is 'Created by Gianpiero Costantino' with a link to 'http://www.gianpierocostantino.eu' and a 'Contact the developer' link. The main content area shows two endpoints under the heading 'dga-service-implementation : DGA Service Implementation':

- POST /v1/detectDGA**: Run the detectDGA Analytic that takes domain-names and check if they are DGA using a third-party algorithm
- POST /v1/matchDGA**: Run the shareDGA Analytic that takes domain-names and check if they are DGA using a public source of known DGAs

Figure 49: Swagger representation for Monitoring of Domain Generation Algorithm DNS-request

6.1.4.1.1. Link to Source Code

<https://devC3ISP.iit.cnr.it:8443/c3isp-wp2/monitoring-dga.git>

6.1.4.1.2. Source Code Description

The Monitoring of Domain Generation Algorithm DNS-request analytic has been developed as API as part of an Eclipse project. In Figure 50, it is shown the analytic hierarchy of the analytic shown as an Eclipse project, the source code of the most important functionalities of the API is available in the DGAServiceImplementation.java object.

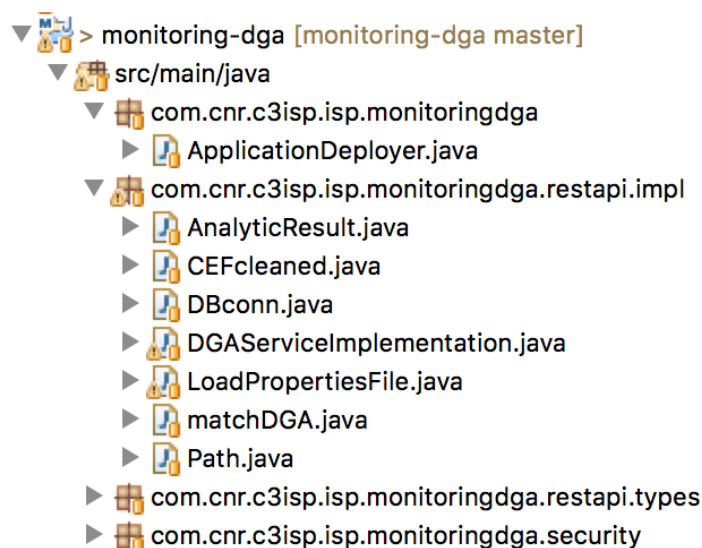


Figure 50: Monitoring of Domain Generation Algorithm DNS-request project and main objects

Source Code of /detectDGA API.

This API is exposed as a POST method that takes as input a JSON file containing the URI to the CEF file stored in the VDL. In particular, the CEF file contains DNS request log and an internal algorithm checks if each request was a DGA. For all DGAs found, the algorithm creates a CEF file.

The following code represents the signature of the API:

```
@RequestMapping(value = "/detectDGA",
consumes = { "application/json" },
produces = { "application/json" },
method = RequestMethod.POST)
public AnalyticResult detectDGA(@RequestBody Path pathVDLParam)
```

When invoked the API requires that the caller specifies where is located the DNS request log file to be processed temporarily stored in the virtual data lake (VDL). The input file containing the DNS requests is already translated into the CEF format and the analytic is able to process. So, for each request written in CEF, the API extrapolates the corresponded domain name. An example of request log written in CEF is in Table 11:

Table 11 - BIND DNS requests written in CEF

CEF messages
CEF:0 DNS_Vendor DNS_CED 1.0 100 DNS query 5 src=192.168.1.2 spt=37239 msg=www.google.com IN A -EDC (192.168.1.9) end=1505484703431 dtz=Europe/Berlin
CEF:0 DNS_Vendor DNS_CED 1.0 100 DNS query 5 src=192.168.1.3 spt=27203 msg=anevmtpra.info IN A + (192.168.1.9)end=1505484704474 dtz=Europe/Berlin

The corresponding source text generated by BIND is:

```
15-Sep-2017 16:11:43.431 client 192.168.1.2#37239 (www.google.com) :
query: www.google.com IN A -EDC (192.168.1.9)
15-Sep-2017 16:11:44.474 client 192.168.1.3#57203 (anevmtprova.info) :
query: anevmtpra.info IN A + (192.168.1.9)
```

To parse the CEF messages, this API uses the following open sources libraries:

```
import com.github.jcustenborder.cef.CEFParserFactory;
import com.github.jcustenborder.cef.CEFParser;
import com.github.jcustenborder.cef.Message;
```

The `/detectDGA` API parses each message with the aim of extrapolating from the `msg` field the domain name resolved by the client that made the request. After, getting out the domain name, for instance `anevmtpra.info`, the analytic invokes a third-party script written in python to discover for DGA based on *ngram* analysis with trained Markov chain mode. This python script is released as open source and some modifications to the source code were required to allow the script to work with a list of domain name available and to generate a corresponding CEF file with all DGAs found. At the end, the CEF file is taken and a new DPO is created in the DPOS.

Source Code of `/matchDGA` API.

This API behaves similarly to the `/detectDGA`. In fact, it takes as input a CEF file containing all domain name requested that have been generated using the BIND DNS software. Even this API is exposed as POST and takes as input a JSON object that specifies where the CEF file is located in the virtual datalake. The signature method of this analytic is represented in the next box:

```
@RequestMapping(value = "/matchDGA",
consumes = { "application/json" },
produces = { "application/json" },
method = RequestMethod.POST)
public AnalyticResult matchDGA(@RequestBody Path pathVdlParam)
```

Once the */matchDGA* is invoked, it receives as input a CEF file as that one depicted in Table 11. Then each line is parsed and the domain name is extrapolated in the same manner as explained above. However, this time the domain name is used to be matched with a table stored with a MySQL database. In particular, the table stores all recent DGA public available through the public source at the url: <https://osint.bambenekconsulting.com/feeds/dga-feed.txt>.

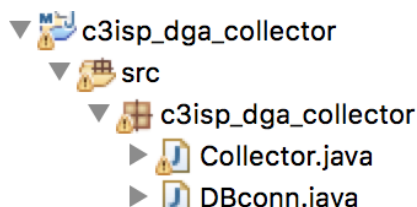


Figure 51: Eclipse project for the script to retrieve public DGAs

To this purpose, an additional script was written to daily download the list and populate the MySQL table. So, when the *matchDGA* queries the table, if the matched domain-name is found in the table then the domain-name is set as DGA. So, at the end of the match all DGAs found in the input files are stored in a new CTI with the CEF format and sent to the ISI as a new DPO.

6.1.4.2. Detection of brute force/DDoS attacks.

At m24 this analytic has not been added yet. As it will be described in D2.3 available for m30, the Detection of brute force/DDoS attacks will analyse service logs to detect brute force or DDoS attacks.

6.1.4.3. Spam E-mail Analysis

The analytic for Spam e-mail analysis has been the first analytic made available through the IAI API. At M24 the Spam E-mail Analysis is made by two API call available through the IAI APIs fully implemented and integrated with the CERT pilot. A third analytic has been added and is currently under development. The three analytics are reported in Figure 52.

POST	/v1/spamEmailClassify	spamEmailClassify
POST	/v1/spamEmailClusterer	spamEmailClusterer
POST	/v1/spamEmailDetect	spamEmailDetect

Figure 52: IAI API calls of analytics for SPAM e-mail analysis

- The *spamEmailClassify* takes as input a set of *eml* files and assigns to them a class based on the spammer goal. The analytics extract first a set of structural features converted in a vector of numerical features to be used as input for the classifier. The full list of features is reported in Figure 53. The classifier will assign to each vector one of the following classes: (i) Advertisement: E-mail used for unsolicited advertisement of products; (ii) Phishing: E-mail used for attempting to steal user credentials on some services; (iii) Malware: E-mail used as a vector for a malicious software to infect the receiving matching; (iv) Portal: E-mail generally showing a

single link that redirects the user on a portal of different categories of products for sale. It is similar to advertisement but makes it harder to assign responsibilities; (v) Confidential Trick: E-mails that attempt to trick the user into paying an amount of money in exchange of a fake service. This function has been fully implemented

Attribute	Description
RecipientNumber	Number of recipients addresses.
NumberOfLinks	Total links in email text.
NumberOfIPBasedLinks	Links shown as an IP address.
NumberOfMismatchingLinks	Links with a text different from the real link.
NumberOfDomainsInLinks	Number of domains in links.
AvgDotsPerLink	Average number of dots in link in text.
NumberOfLinksWithAt	Number of links containing “@”.
NumberOfLinksWithHex	Number of links containing hex chars.
SubjectLanguage	Language of the subject.
NumberOfNonAsciiLinks	Number of links with non-ASCII chars.
IsHtml	True if the mail contains html tags.
EmailSize	The email size, including attachments.
Language	Email language.
AttachmentNumber	Number of attachments.
AttachmentSize	Total size of email attachments.
AttachmentType	File type of the biggest attachment.
WordsInSubject	Number of words in subject.
CharsInSubject	Number of chars in subject.
ReOrFwdInSubject	True if subject contains “Re” or “Fwd”.
NonAsciiCharsInSubject	Number of non ASCII chars in subject.
ImagesNumber	Number of images in the email text.

Figure 53: Features for e-mail analysis

- The *spamEmailClusterer* exploits the CCTree [18] algorithm to separate spam e-mails in campaigns, exploiting structural similarity. Campaigns are a set of spam e-mails with a similar structure, generally generated by the same spammer. The input is a set of *eml* files from which are extracted the same features exploited by the *spamEmailClassify* analytic. This analytic at M24 is fully implemented and integrated in the CERT pilot.
- The *spamEmailDetect* analytic takes as input a set of e-mails and separates them into genuine (ham) e-mails and unsolicited ones (spam). The analytic at month 24 has been developed but has still to be integrated.

Source Code for Spam Analysis

The code of spam analysis is part of the IAI API project shown in Figure 54.

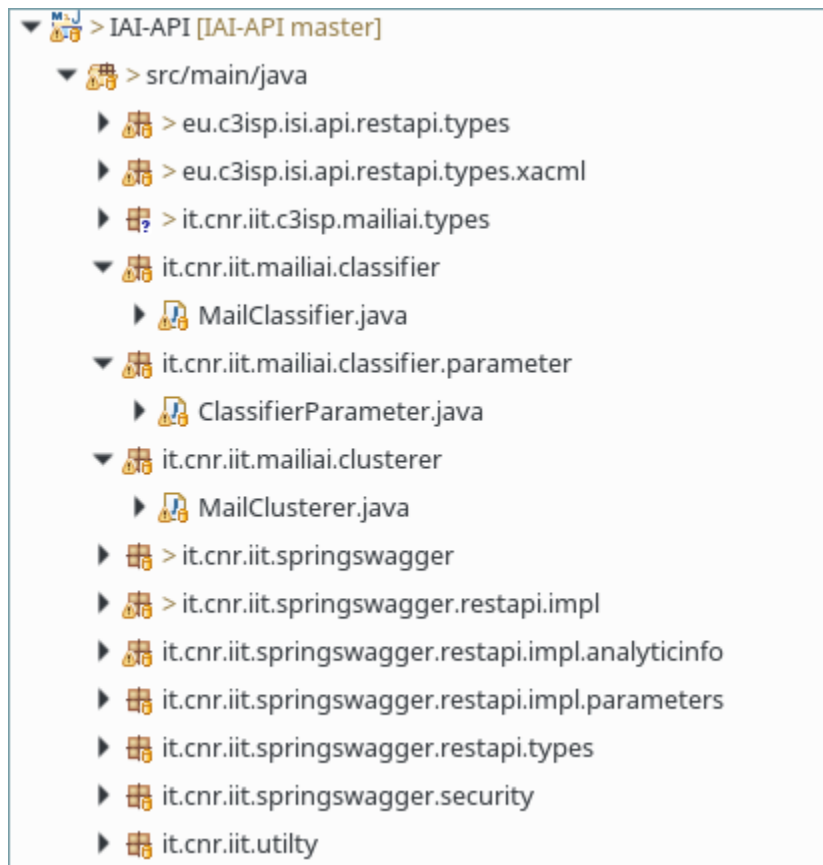


Figure 54: IAI API code structure

The functions are implemented as Java classes, invoked directly by the IAI API java code when triggered via REST. The `MailClassifier.java` class contains the function for classification, including the primitives to train the classifiers or add new knowledge. It is also possible to select the classifier as a parameter. By default, the used classifier is the *CCTreeClassifier* [19]. The `MailClusterer.java` includes instead the primitives to perform the clustering exploiting the *CCTree* algorithm, and the primitives to compute the quality of generated clusters in terms of purity. The `ClassifierParameter.java` includes the functionalities to extract features from *eml* and generate the vector of parameter used as input for classifiers.

7. Visualization of Security Analytics

7.1. Component description

The SATURN Visual Analytics tool will be used to provide visualisation of security insights extracted from a multitude of security data (CTI) that have been shared within C3ISP platform as well as from results produced by C3ISP-own collaborative analytics functions, such as list of identified DGA (Domain Generation Algorithm) domain names or malicious source hosts. In-depth and interactive analysis of the data will be performed by means of ten different types of inter-linked visual gadgets, e.g. bubble chart, link-analysis, trends, clustering, geographical.

7.2. Maturation status

SATURN already has TRL 9 status and is regarded as a *Legacy Analytics Engine* within the C3ISP IAI subsystem. No further development of new functions was planned to mature the product further. However, maturation towards its integration into the C3ISP architecture was needed to enhance the analytics capability of C3ISP-powered cyber sharing platform on the one hand, and to test and validate the flexibility of the C3ISP reference architecture in accommodating existent and legacy technologies on the other hand. The specific tasks that have been completed by M24 towards such maturation are summarised as follows:

- SATURN software needs to be deployed within the protected BT network environment due to its licensing agreement. In contrast to other C3ISP components, the tool is therefore not accessible from the public Internet. To overcome this issue a VPN solution was implemented using the OpenVPN software for establishing a secure tunnel between the respective BT network gateway and potential SATURN user's machine. As described in D7.3 a semi-permanent VPN tunnel has also been set up between the BT gateway and the IAI machine to allow HTTP redirection service which would remove the requirement of each client machine for setting up their own VPN tunnel. This also allows SATURN to securely access the data stored within the C3ISP sub system component (i.e. Virtual Data Lake).
- SATURN will read the data from a Virtual Data Lake (VDL) instance prepared by the C3ISP IAI component. As described in D7.3 a VDL implementation using MySQL database system has been completed. We have successfully verified that a VDL-URI provided by the corresponding C3ISP component which contains the unique MySQL database connection details and access credentials, can be used to create a new data source within SATURN. The configuration has been done manually using SATURN's user interface to connect to the database and associated table as well as to map the table columns to respective data attributes for further analysis in SATURN.
- In order to strive for more seamless integration of SATURN we looked at options to automate its data source configuration and mapping without the need for user interaction. We have verified that this can be achieved under certain circumstances by applying our internal knowledge of SATURN. Due to its confidential nature, the details cannot be disclosed here.
- Another aspect of integration is the identity management comprising user authentication and authorization. Being able to use a common user identity and role across different C3ISP components is beneficial in order to remove unnecessary complexity for controlling access to various resources such as CTI data or API function calls. The C3ISP testbed is currently adopting OpenLDAP as its Identity

Manager. We have configured SATURN to use C3ISP’s OpenLDAP server to manage its user base instead of using SATURN’s own authentication module. Each SATURN user needs to be assigned a role either as *User* or *Administrator*. This should be reflected in the group membership settings on the OpenLDAP server. We have created the required LDAP directory structure for some test users and stakeholder groups related to the C3ISP Enterprise Pilot. In particular, we grouped the users into two company groups (more groups can be added later as needed). Additionally, each user is assigned a stakeholder group, i.e. *SecurityAnalyst*, *Security Operation Executive*, or *Data Policy Officer*. All these users will be recognized by SATURN to have the role *User*. Another user is assigned a stakeholder group *Development Manager* which will be recognized by SATURN as its *Administrator* user. We have successfully tested the SATURN’s LDAP configuration within the context of Enterprise Pilot. LDAP directory structure for the other C3ISP pilots (i.e. ISP, CERT, and SME Pilots) can later be created as necessary.

7.3. Requirement Analysis at M24

Table 12 – Visualization of Security Analytics Requirements Status

ID	MET	Description
C3ISP-Com-Usa-001	YES	The tool offers a rich and intuitive graphical user interface to allow users to carry out different tasks, starting from configuring data sources, event attributes mapping, exploring and filtering the datasets, up to visualising the data using a variety of visual gadgets.
C3ISP-Com-Usa-002	YES	The tool supports two ways for data ingestion. Data can be read from databases, i.e. RDBMS (Oracle, PostgreSQL, MySQL, SQL server), columnar database (Vertica), and Hadoop cluster (via Cloudera Impala), or uploaded from files in Excel and CSV formats.
C3ISP-Com-Usa-003	YES	The tool provides an “Explore” component which is capable of handling billions of events at a time and may be used to construct more specific, complex database queries for further analysis. In total there are 10 different graph views, such as bar or line charts, mini graph view, parallel coordinates, etc., which can be used to help users visually filter and explore the data.
C3ISP-Com-Usa-004	YES	The tool provides an “Analyse” component which allows for fine-grained visualization and analysis of the data read into the system. It consists of one or more ‘Gadget’ windows which can themselves be configured to show one of several different visualizations such as bubble chart, trends, network, radial, pie chart, geolocation map, etc. It also supports unsupervised clustering algorithm and visualization to elicit patterns of interest in the data.

7.4. *First release of the component*

SATURN has TRL9 status and is currently deployed within protected BT network environment. The implemented functionalities have been described in D8.1 and all the requirements have already been met. In the next phase, further efforts will focus on integrating the tool with other C3ISP components as needed.

7.4.1.1. Link to Source Code

Since SATURN is not an open-source software, its source codes and binaries cannot be made available to public.

7.4.1.2. Source Code Description

Since SATURN is not an open-source software, its source codes and binaries cannot be made available to public.

8. Anonymization and Homomorphic Encryption Algorithms

Privacy-preserving analytics within C3ISP are enabled by anonymization of sensitive data, provided by SAP, and data encryption that allows to perform operations on the encrypted data in the form of homomorphic encryption, provided by CEA.

Anonymization alters the data in a way that reduces the risk to identify any individual contained in the dataset by removing, masking or randomizing personally identifiable information within the data and enables an analyst to work on the anonymized data in the clear. Homomorphic encryption, on the other hand, conceals the original data from the analyst but enables computation on the ciphertext (the product of the encryption) such that the computation product is another ciphertext containing the desired result. In the context of C3ISP this enables participants to share their own data which was previously not possible due to privacy concerns (regarding information in the non-anonymized data) and to outsource computation to an untrusted server.

8.1. Anonymization Algorithms

8.1.1. Component description

The Anonymization Toolbox component is a research prototype developed by SAP. The provided anonymization methods include the exclusion of personally identifiable information (anonymization by suppression, e.g., removal of unique ids and names), the removal or masking of parts of the data (anonymization by generalization, e.g., randomizing the subnet of an IP address) and incorporating fine-tuned noise (anonymization by perturbation, e.g., adding noise to counts or locations).

The latter method refers to a new area of research for a privacy definition called *Differential Privacy*[4],[5] which protects individuals in the data but still enables statistical information about a population (group of individuals) to be obtained. Roughly speaking, random noise is sampled from a distribution (e.g., Laplace) parametrized in such a way as to add as little noise as needed to hide any *single* individual value for a given analytics function and yet have almost no noise when looking at aggregates of individuals (e.g., the mean of noisy values contains almost no noise, i.e., has low variance compared to noisy individual values).

How to define mechanisms that add *just enough* noise for a given analytical function and yet maintain utility, i.e., usefulness of the result, is an active and on-going area of research. We analysed the possibility to reduce the needed noise in the context of outlier detection and published a paper at DBSEC[6]: Individuals are separated in two groups, non-attackers (which we seek to protect) and attackers (which we seek to find), whereas the latter are presumed to consume much more resources than the first – as large resource consumption might indicate a malfunction. In the context of C3ISP this could mean a DDoS attack (many more requests than normal) or SPAM attack (much more e-mails/bytes sent than usual). We aim to further investigate how to carefully select the noise for meaningful analysis in the context of cyber threat information sharing.

8.1.2. Maturation status

The existing anonymization techniques (removal/suppression, generalization, perturbation) were extended and tailored especially for use within C3ISP. For one, we simplified the intricate parameter selection for the differentially private noise distribution to enable non-expert users to select anonymization methods with pre-defined parameters. Thus, the DSA Editor can display simple, pre-configured anonymization suggestions for the user. For

example, the method to anonymize via addition of Laplace distributed noise (Laplace mechanism [5]) was simplified to protect counts for mean for three predefined levels of privacy as *LOW*, *MEDIUM*, *HIGH* (previously it required the selection of a privacy parameter and sensitivity parameter, i.e., the largest absolute difference the inclusion/exclusion of any individual can have on the function evaluation).

Furthermore, a distinction which part of a delimited string is to be anonymized was added, i.e., if the removal should begin from the leftmost or rightmost part of a string or if the lower or upper part of an IP address should be anonymized. Also, IPv4 address pseudonymization was added. Instead of removing or masking IPv4 addresses (upper or lower) bits are replaced by random bits to maintain the structure of IPv4 addresses for further processing or analytics that may parse logs via regular expressions.

The exponential mechanism [8] is investigated as potential maturation. Instead of perturbing individual values (or the function result computed with these values) the exponential mechanism defines a selection probability for all possible values (e.g., all 32-bit integers) for a given evaluation function (e.g., median) with regards to a given dataset (e.g., dataset {3,5,5,1023}). The exponential mechanism outputs a value that is exponentially more likely to be close to the actual function result (e.g., median here is 5) than all other possible values. It fulfils the definition of Differential Privacy and therefore provides a strong privacy guarantee and can be applied to different functions. (However, as a non-zero probability for *all* possible values must be computed with regards to a potentially large dataset it can be computationally inefficient in certain cases.) The median is an important statistical measure that unlike the mean is robust (outliers do not easily skew the result) and is an example for a representative element from a dataset. It can be used to find, e.g., the median number of connections, or bytes sent/receives, etc., in a privacy-preserving fashion and can be the basis for further analysis. E.g., by first finding the (privacy-preserving) median for normal users and then scaling/reducing the differentially private noise to protect only normal users and not outliers (presumed attackers) as suggested in [6] [6].

The Toolbox was extended to run as-a-Service, i.e., instead of configuring required data access parameters/credentials (MySQL database URL, username, password) in a file they are now submitted to the Toolbox as additional parameters per request and database connections are established when needed and not as a connection pool.

8.1.3. Requirement Analysis at M24

ID	MET	Description
C3ISP-Com-AA-001	PARTIALLY	Parameters for simplified usage of the component were defined and a list of currently supported methods can be found in Section 8.1.4. Improvements of existing methods or potential extensions are being investigated.
C3ISP-Com-AA-002	YES	The input data type (the component supports “text”, i.e., strings and numbers) is implicitly defined via the method (e.g., <code>anonymiseDelimitedStringsByRemoval</code> expects, as the name implies, a string). The input for <code>anonymiseByRemoval</code> is treated as a string.

C3ISP-Com-AA-003	YES	The current version has definitions for anonymization methods and parameters based on analytical treatment as required and a list of currently supported methods can be found in Section 7.1.4.
C3ISP-Com-AA-004	YES	An interface for privacy-preserving operations is provided via XML requests and a first version of a REST API.

8.1.4. First release of the component

The Toolbox runs as a Web Service on an Apache Tomcat server, listens on port 8080 and expects XML as input format (as described in more detail in Deliverable D8.1). The data is expected to be provided as a MySQL database by the **BufferManager** component.

To facilitate the communication between C3ISP components (i.e., the DMO engine) and the Toolbox and to conform to the REST API design within the C3ISP framework the following API was defined and a translation between these API parameters (currently JSON) and the ones expected by the Toolbox (XML) will be performed.

Method Name	Note	Parameter Example
<i>/v1/anonymiseByRemoval</i>	The entire column is removed (anonymization by suppression), e.g., for personal identifiers such as ID numbers, full name, etc.	{ "column": "string" }
<i>/v1/anonymiseCountsWithNoiseForMean/</i>	Adds Laplace distributed noise with sensitivity 1 to guarantee Differential Privacy, where sensitivity is the maximum impact that the inclusion/exclusion of an individual can have on a function evaluation.	{ "column": "string", "dataAccess": { "dbpassword": "string", "dburl": "string", "dbuser": "string" }, "protectionLevel": "LOW" }
<i>/v1/anonymiseDelimitedStringsByRemoval/</i>	Removes substrings up to a delimiter (e.g. '.' for IPv4 addresses, ':' for MAC/IPv6, '@' for e-mail); parameter 'area' defines if removal affects 'lower' parts (removal begins from the right) or 'upper' parts (begins from left).	{ "area": "LOWER", "column": "string", "dataAccess": { "dbpassword": "string", "dburl": "string", "dbuser": "string" }, "delimiter": "IPV4", "protectionLevel": "LOW" }
<i>/v1/anonymiseIpsByRandomization/</i>	Randomizes upper or lower bits (see parameter 'area') of IPv4 addresses (number bits, i.e. netmask, depends on protection level).	{ "area": "LOWER", "column": "string", "dataAccess": { "dbpassword": "string", "dburl": "string", }

		<pre>"dbuser": "string" }, "protectionLevel": "LOW" }</pre>
<i>/v1/anonymiseLocations/</i>	<p>Provides Geo-Indistinguishability [7] for locations x,y (informally, it is more likely that closer points remain similar, far away points remain distant; more formally, $\text{distributionDistance}(M(x), M(y)) \leq \text{epsilon} \times r$ where $\text{euclidDistance}(x,y) \leq r$, M is the Geo-Ind. mechanism, and $\text{epsilon} \times r$ defined via protectionLevel parameter).</p>	<pre>{ "column": "string", "dataAccess": { "dbpassword": "string", "dburl": "string", "dbuser": "string" }, "protectionLevel": "LOW" }</pre>
<i>/v1/anonymiseMultipleColumns/</i>	<p>Define anonymization method and parameters per column. The method names are the API method names likewise for parameters.</p>	<pre>{ "anonConfigList": [{ "area": "LOWER", "column": "string", "delimiter": "IPV4", "method": "anonymiseDelimitedStringsByRemoval", "protectionLevel": "LOW" }], "dataAccess": { "dbpassword": "string", "dburl": "string", "dbuser": "string" } }</pre>

Each function anonymizes a single column (or “field” as it is called for the Common Event Format (CFE)) and is provided to give an overview of existing methods and a short description of their usage including required parameters. In case multiple columns need to be anonymized *anonymiseMultipleColumns* takes as input a list of columns to be anonymized and their respective anonymization methods and parameters. (Thus, *anonymiseMultipleColumns* can provide all anonymization methods and could replace the methods operating on single columns, however, this reduces the readability of the API.)

The current API is expected to change, either by extension for further techniques/parameters, by refinement of existing methods or further simplifications needed for the DSA Editor.

8.1.4.1. Link to Source Code

<https://devc3isp.iit.cnr.it:8443/c3isp-wp8/isi/dsa-adapter/anonymization-toolbox/anonymization-toolbox> (binaries)

8.1.4.2. Source Code Description

A list of the APIs and their description is reported in Section 8.1.4. The source code components for the Toolbox integration can be seen in Figure 55:

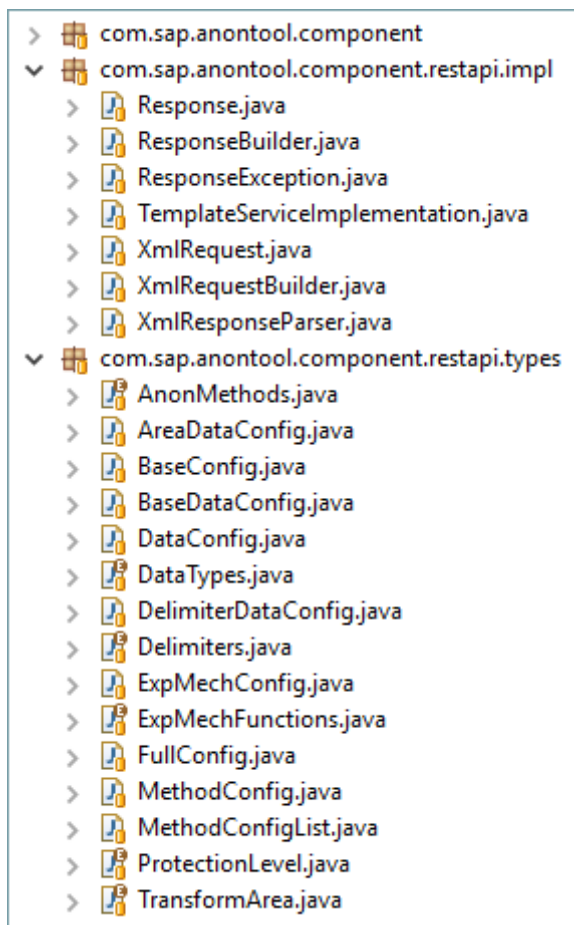


Figure 55: Anonymization Toolbox integration code structure

The implementation for the REST API includes XmlRequestBuilder and XmlResponseParser as an interface to the Anonymization Toolbox. The supported anonymization methods require at least a column to be anonymized, a protection level, data access credentials and the name for the input and output table that will contain the anonymized version of the input table.

An overview of all anonymization methods of the REST API is defined in AnonMethods.java, including the DMO options, i.e., which parameters are set for a given protection level and anonymization method. For example, the DMO action UPPER_NETMASK_LOW_PRIVACY corresponds to *anonymiseIpsByRandomization* with parameter "area": "LOWER" and "low privacy" corresponds to only randomizing one octet. Appendix 1 describes the DMO parameters and options.

8.2. Homomorphic Encryption Algorithms

8.2.1. Component description

The Fully Homomorphic Encryption (FHE) Analytics component is developed by CEA. It aims at performing specific data analytics, all the while preserving data confidentiality. This is done using hybrid encryption. It makes use of two types of cryptosystems: Brakerski/Fan-

Vercauteren (B/FV) *homomorphic cryptosystem* to perform analytics on encrypted data (ciphertext) and Kreyvium symmetric cryptosystem to permit prosumer to encrypt and send ciphertexts more efficiently using *transcription*¹⁴ mechanism [9].

8.2.2. Maturation status

The FHE Analytics employs in the backend, Cingulata. It is a compiler toolchain and runtime environment, developed by CEA, for running C++ programs over encrypted data by means of fully homomorphic encryption techniques.

Cingulata became a Free Open-Source Software (FOSS) in Y2. The tool is available on the website <https://github.com/CEA-LIST/Cingulata>.

What follows explain the work done during Y2 to maintain confidentiality and deliver new functionalities useful to C3ISP project.

Data confidentiality

Confidentiality is one of the major missions in the FHE Analytics component. It aims at protecting secret information against adversaries. The security level of a cryptosystem instance is a quantity which enables to estimate the number of binary operations needed to break the cryptosystem (i.e. decrypt without knowing the secret key) against known attacks. There exist several families of attacks and currently, there is no best attack on B/FV cryptosystem [15]. Best attack depends on the parameter sets used to instantiate the cryptosystem. In addition, there is no consensus on the cost estimation of an attack, there exist at least 14 different cost models in the literature¹⁵ which lead to different conclusions. Homomorphic cryptosystems such as B/FV [11] rely on difficult mathematical problems on lattices. In the current version of Cingulata, the implementation of B/FV use, by default, parameter sets obtained following the recommendations given in the article introducing the B/FV cryptosystem to get a security level of 128-bit. Optimal parameter selection in lattice-based cryptography is an open problem [12][17]. A tool, called the *lwe-estimator*¹⁶[17][13] have been developed to help the community to choose secure parameter set. It is regularly updated to integrate new results. We employed it to check the security level of the parameter set used in Cingulata implementation. In March 2018, we identified that an attack called Primal-uSVP is estimated feasible against the parameter sets used in Cingulata. This result follows a work presented in Asiacrypt 2017 which gives more precise results to estimate the cost of Primal-USVP attack[14]. To maintain confidentiality with 128-bit of security, we then developed a program which makes use of *ChooseParam* algorithm in [13][13] and on the more recent version of the *lwe-estimator* to determine secure parameter sets against a family of attacks such as Primal-uSVP to solve LWE (Learning with Error [16]) hard mathematical problem on which the security of B/FV is based. Rather than using by default a fixed parameter set susceptible not be secure anymore in few months if better attacks are proposed or more pessimistic cost attack estimation are obtained, we store and update a database containing estimated-secure parameter sets. From a high-level view, a parameter set depends on the multiplicative depth of the Boolean circuit that we apply on encrypted data. To recapitulate, this parameter is the maximal number of multiplications between an input and an output of the Boolean circuit. We restrict our attention to four popular cost models among the one used by lattice-based cryptosystem candidates in ongoing NIST Post-Quantum

¹⁴ For short, transcription permits to transform symmetric ciphertexts into homomorphic ciphertexts.

¹⁵<https://estimate-all-the-lwe-ntnu-schemes.github.io>

¹⁶ LWE stands for Learning with Errors, it is the name of a difficult cryptographic problem with lattices.

Cryptography (PQC) Standardization¹⁷ project. Each cost model gives different security estimates. Considering them, we compute and provide up-to-date secure parameter sets for multiplicative depth between 4 and 20 which is the suitable range for B/FV cryptosystem employed to evaluate C3ISP data analytics over ciphertexts.

Data Analytics

The FHE Analytics permits to operate on a fixed-length string (such as IPv4). In addition, variable-length string (such as words) can be treated. To enable this, a string hash function¹⁸ is applied during precomputation. This operation maps variable-size data to fixed-size data called **hashes**. These hashes are 32-bit values. Current use-case in C3ISP makes use of set of IPv4 addresses with homomorphic algorithm (circuit) membership. An IP set is an IP list with no duplicate. It is possible to manage lists with duplicate elements but not desirable. The circuit *multiplicity* operates on a list rather than a set. It is much costlier nevertheless it also allows to count the number of occurrences of an IPv4 in a list. This could also be used to detect suspicious number of connections. These two circuits can be used in combination with a *transcription* circuit to decrease prosumer charge.

Performance depends mainly on multiplicative depth and on number of IPv4s for circuits. The multiplicative depths are given in Table 1. The two circuits have been precomputed for different list size. In addition, each homomorphic algorithm has different prerequisites given in the Tables 2 and 3.

Table 13 - Circuit multiplicative depth to detect a suspicious IPv4 in a blacklist. The bigger is the slower.

	Set of IPv4 (no duplicates)	List of IPv4s (possible duplicates)
Without transcription	5	12
With transcription from Kreyvium cryptosystem	12	19

8.2.3. Requirement Analysis at M24

ID	MET	Description
C3ISP-Com-HE-001	YES	We represent IPv4s addresses in Cingulata as one 32-bit integer rather than four 8-bit integers.
C3ISP-Com-HE-002	YES	We test membership to a list (e.g., a list of malicious IPs) over ciphertexts with Cingulata. There are two cases to distinguish in terms of performance. It depends if the list contains duplicate elements (circuit multiplicity) or not (circuit membership).

¹⁷<https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization>

¹⁸<http://www.cse.yorku.ca/~oz/hash.html>

	PARTIALLY	We compute the intersection of two lists with Cingulata
C3ISP-Com-HE-003	PARTIALLY	List size impacts time and memory requirements. Managing big lists require additional (pre/post) computations on the client side. We employ transcription mechanism to decrease computation and communication costs on client side.
C3ISP-Com-HE-004	PARTIALLY	We use precomputation to represent IPs addresses as integers of fixed size in Cingulata. Other data types would require other precomputations before data encoding such as hashing for variable-length data. Encoded values can be integers of size 8, 16, 32 or 64 bits.

8.2.4. First release of the component

The first release of the FHE Analytics component permits to test if an encrypted IPv4 belongs to a set of encrypted IPv4s contained in a blacklist. It serves to determine if a connection is malicious or not. To proceed, it takes as input one IP address and compares it homomorphically to each IP stored in the considered blacklist. Each IP is encrypted bitwise.

Parameter selection stays a difficult problem in lattice-based cryptography [12]. This field is explored by the research community for different reasons. One of them is that lattice-based cryptography enables new applications such as homomorphic cryptography. Performance is a key question from an application perspective. Homomorphic cryptography suffers from low-performance in terms of time, memory, communications costs compared to a traditional one. We benchmark the data analytics on the iaic3isp.iit.cnr.it server with current parameter sets. As mentioned above, parameter sets will be automatically updated using lwe-estimator tool. Parameter selection is essential as it impacts both security and performance.

To improve efficiency, we employ transcription (or transciphering) techniques by interacting with Key and Encryption Manager. The process is described in D7.3, Section 7.2. Transcription makes use of two cryptosystems during the homomorphic computation protocol instead of one. The first one is low-cost (homomorphic-friendly), and the other one is homomorphic. Transcription saves time, memory and bandwidth consumption during the phase before the homomorphic computation. This phase consists in encrypting confidential data on the prosumer side and send them to the FHE Analytics. Instead of directly encrypting data with Fan-Vercauteren homomorphic asymmetric cryptosystem, the prosumer encrypts them using Kreyvium homomorphic-friendly symmetric cryptosystem. Symmetric encryption offers the advantage to be faster than asymmetric one. In addition, ciphertexts (encrypted data) sent to the FHE Analytics using Kreyvium cryptosystem, are much smaller. In return, the FHE Analytics server has additional work. It has to transcribe the received ciphertext that is to compute a Fan-Vercauteren ciphertext from a Kreyvium ciphertext to enable homomorphic computations. The homomorphic circuit to process has multiplicative depth 7. As said above, Fan-Vercauteren permits, in practice, at most to treat homomorphic circuit of depth around 20. This means that the homomorphic algorithms, we employ on data have to be described by a Boolean circuit having a multiplicative depth of 13, when using transcription technique.

Table 14 - Circuit membership to detect malicious IPv4s in a set of IPv4s

	Description
--	-------------

<i>Circuit Fullname</i>	c3isp-membership	It determines the presence or absence of an IP in a set of IPs.
<i>Circuit Inputs</i>	(1+ <i>size</i>) variables of type Integer32	One target IP plus <i>size</i> IP in the blacklist, where <i>size</i> is the number of considered IP
<i>Circuit Output</i>	1 Bit	Answer is true or false
<i>Multiplicative Depth</i>	5	Impacts performance. Smaller is better. Independent of set size.
<i>Max number of IPv4s</i>	600	The circuit depends on the number of considered IPs. One circuit has been precomputed per each size less than 600.
<i>Required parameter</i>	Target IP	The operations aims at determining if target IP is in the considered set.
<i>Required parameter</i>	Number of IP considered in the set	Impacts performance. Smaller is better. Needed to define which circuit to choose in the database.
<i>Optional parameter</i>	Set name	Default value: ipv4.dat Blacklists are in the directory: /home/hcatworks/fhe/instance/membership/blacklists
<i>Prerequisite</i>	No duplicate IP in the list	A circuit without this constraint would be affected in terms of performance. We should favour the circuit multiplicity if this constraint cannot be respected.
<i>Prerequisite</i>	The list must contain the number of IPs given in parameter	It can contain more IPs but not less.

Table 15 - Circuit multiplicity to detect malicious IPv4s in a list of IPv4s

		Description
<i>Circuit Fullname</i>	c3isp-multiplicity	It counts the number of occurrence if an IP in a list of IP and thus permit to decide if an IP belong to a blacklist or not.
<i>Circuit Inputs</i>	(1+ <i>size</i>) Integer32	One target IP plus <i>size</i> IP in the blacklist, where <i>size</i> is the number of considered IP
<i>Circuit Output</i>	1 Integer8	The number of occurrences of target IP in the list
<i>Required parameter</i>	Target IP	The operations aims at determining how many times target IP appears in the considered list.
<i>Required parameter</i>	Number of IP considered in the set	Impacts performance. Needed to define which circuit to choose in the database.

<i>Optional parameter</i>	List name	Default value: ipv4.dat Blacklists are in the directory: /home/hcatworks/fhe/instance/multiplicity/blacklists
<i>Prerequisite</i>	The two lists must contain the number of IPs given in parameter	They can contain more IPs but not less.

Below, the Figures indicate time and memory performance for the two circuits with list of size 100. The original parameters offer an estimated security level λ of 35 bits. Security must be at least 100 bits and should be at least 128 bits according to the French Network and Information Security Agency (ANSSI)¹⁹. The first column indicates three input parameters needed to estimate the security:

1. The multiplicative depth of the circuit (5 for membership and 12 for multiplicity)
2. The chosen cost model. There are different ways to estimate the security, some of them are optimistic (BKZ sieve), and some other are pessimistic (Q-Core sieve). There is no consensus in the community for one model. We consider the 4 most used models in the ongoing Post-Quantum Cryptography NIST conference (BKZ enum, BKZ sieve, Core Sieve, Q-Core Sieve).
3. The desired security level. We follow ANSSI recommendation to obtain parameters which offer a security level greater than 128.

The output parameters n and q permits to define the ciphertext space of polynomials of degree n with coefficients modulo q . The bigger they are, the bigger is the memory used. We indicate time performance with and without precomputation. IPv4 is represented with 4 integers between 0 and 255 (8-bit integer). The precomputation consists in representing them as one 32-bit Integer. The associated circuit contains less AND gates. We remind that the multiplicative depth and the number of AND gates are the most important parameters to minimize execution time when using Brakerski/Fan-Vercauteren scheme.

Last column indicates memory usage, more precisely encrypted bit size. An IPv4 corresponds to 32 encrypted bits. The encrypted yes-no answer of the circuit membership corresponds to one encrypted bit. The encrypted integer answer of the circuit multiplicity corresponds to an 8-bit integer (it is at most 255).

circuit membership	n	$\log_2(q)$	λ	time (w/ precomp)	time (w/o precomp)	encrypted bitsize
original parameters	1024	474	35	6 s	8 s	128 KiB
5_bkz_sieve_128	8192	162	180	53 s	60 s	396 KiB
5_core_sieve_128			146			
5_q_core_sieve_128			133			

Figure 56: Updated security λ and performance on circuit membership to test homomorphically if an IPv4 belongs to a set of size 100. Parameters computed using lwe-estimator tool (commit ID=76d05ee)

circuit multiplicity	n	$\log_2(q)$	λ	time (w/ precomp)	time (w/o precomp)	encrypted bitsize
original parameters	1024	2321	35	52s	53 s	590 KiB
12_bkz_sieve_128	32768	377	152	252s	576 s	1.7 MiB

Figure 57: Updated security λ and performance on circuit multiplicity to test homomorphically how many times an IPv4 is in a list of size 100. Parameters computed using lwe-estimator tool (commit ID=76d05ee)

¹⁹ https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf

Time and memory performance are affected by updating security, but these ones are mandatory to offer compromise security requirements (80 bits for most optimistic). The circuit multiplicity can be used in the case of duplicated IP, it means that an IP can have at least two occurrences in the black list. However, this case takes more time and memory for processing. This is mainly due to multiplicative depth gap between the two circuits.

In the following Figure Figure 58Figure 49, it is illustrated the swagger representation on this analytic showing the two APIs



Figure 58: FHE Analysis API

8.2.4.1. Link to Source Code

The code source is available at:

<https://devC3ISP.iit.cnr.it:8443/c3isp-wp2/conn-malicious-host.git>

8.2.4.2. Source Code Description

The list of the APIs and the description of these API is reported in deliverable D7.3, in Section 5.1.2

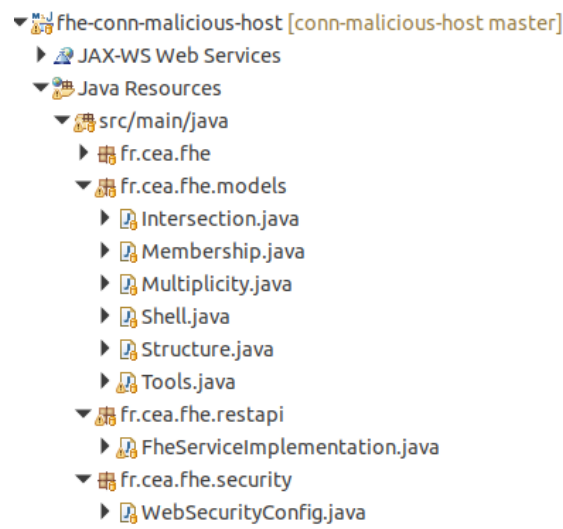


Figure 59: FHE Analysis Toolbox integration code structure

The main package *fr.cea.fhe* contains the *ApplicationDeployer* class, in charge of enabling the Spring framework activation for the web application.

An overview of all homomorphic encryption methods and the analysis models are defined in the package *fr.cea.fhe.models*.

The package *fr.cea.fhe.restapi* contains all the definitions for APIs RestFull service.
Lastly, *fr.cea.fhe.security* contains the Spring security initialization.

9. Managed Security Services

9.1. Component description

In the context of the C3ISP project, the Managed Security Services (MSS) component provided by BT is also known internally as the “Intelligent Protection Service” (IPS). As currently it is only required for use as an MSS in the SME Pilot, it is also referred to as “SME MSS”. However, other C3ISP Pilots can make use of other commercial or open source MSS as per their requirements, including IPS.

IPS enables the protection of systems, applications and data processing on a mix of public and private cloud environments through a collection of security functions that can be offered as managed security services.

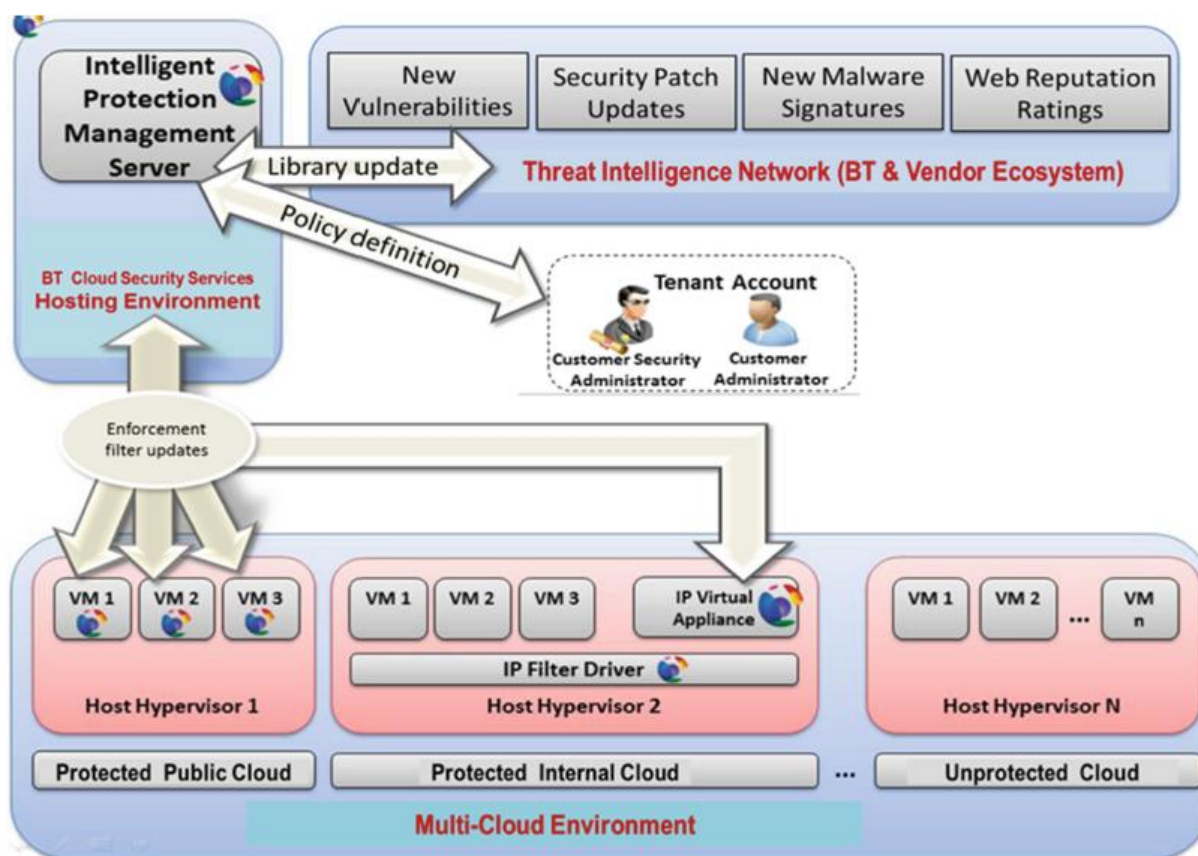


Figure 60: Overview of the IPS high level architecture

An architectural overview of the BT IPS solution offered to each tenant (SME) can be divided into three dimensions as depicted in Figure 60:

- Policy enforcement: this is the mechanism used to manage the protection of a system; in this case, it is an agent installed on a virtual machine or a physical server.
- BT Cloud Security Service: this is the management mechanism used by the IPS for defining security policies based on a library of rules that include virtual patches for a very large number of systems and applications, firewall and protocol rules, etc., and for updating the configuration and enforcement rules of the agents.

- Threat intelligence: this is the mechanism for enhancing the data-base of heuristic rules, attacks or virus signatures, vulnerabilities, etc., via a network that includes a large number of security and application vendors, as well as contributions from BT's security ecosystem.

9.2. Maturation status

BT IPS is part of BT's Managed Cloud Security Products & Solutions portfolio. IPS is a TRL 9 solution and is commercially sold to BT Managed Cloud Security customers as a value-added service. The implemented functionalities have been described in D8.1 and all the requirements have already been met.

9.3. Requirement Analysis at M24

Table 16 – Managed Security Services Requirements Status

ID	MET	Description
C3ISP-COM-REQ-MSS-1	YES	This requirement is currently fulfilled by an instance of the IPS deployed on the BT's Cloud research platform (https://ipserver.zion.bt.co.uk:4119/).
C3ISP-COM-REQ-MSS-2	YES	This requirement is currently fulfilled by an instance of the IPS deployed on the BT's Cloud research platform (https://ipserver.zion.bt.co.uk:4119/).
C3ISP-COM-REQ-MSS-3	YES	This requirement is currently fulfilled by an instance of the IPS deployed on the BT's Cloud research platform (https://ipserver.zion.bt.co.uk:4119/).

9.4. First release of the component

An instance of the BT IPS has been deployed on the BT's Cloud research platform (<https://ipserver.zion.bt.co.uk:4119/>) and the all the SME partners (3D Repo, CHINO and GPS) and UNIKENT have been provisioned with tenant accounts on this service. In the next phase further efforts will focus on integrating the tool with other C3ISP components as needed.

Using an MSS Agent, VMs or physical servers can be connected to the IPS and enable their administrators to remotely monitor and manage the protection of their environment. There are currently two ways of making a VM manageable by IPS:

1. The user can download the agent installer from the IPS, according to the operating system and machine architecture of their VMs or physical servers.
2. The users can be provided an installation and configuration script compatible with the operating system and machine architecture of their VMs or physical servers. On running the script, it automatically contacts the IPS and downloads, registers and activate the appropriate agent software on the VM or the physical server.

9.4.1.1. Link to Source Code

Since BT IPS is not an open-source software, its source codes and binaries cannot be made available to public.

9.4.1.2. Source Code Description

Since BT IPS is not an open-source software, its source codes and binaries cannot be made available to public.

10. Conclusions

This deliverable is the first output of WP8, “C3ISP Data Sharing, Analytics and Crypto Technology Maturation”, due at M24. The main goal of this deliverable is to document the first software release of the components of the C3ISP framework, i.e., of the Information Sharing Infrastructure (ISI) and of the Information Analytics Infrastructure (IAI). These tools and technologies cover most of the functionalities required in the C3ISP project. This deliverable report maturation status and implementation details for each component. The requirement status (defined in D8.1) is updated to reflect the progress status of the requirement at M24.

11. References

- [1] 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, 54
- [2] A. Grigoris and F. Van Harmelen. *Web Ontology Language: OWL*. In Handbook on Ontologies in Information Systems, pages 67–92. Springer, 2003. 18
- [3] <http://csrc.nist.gov/groups/SNS/rbac/>
- [4] Cynthia Dwork. 2006. Differential Privacy. In Automata, Languages and Programming, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.
- [5] Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2014), 211–407.
- [6] Böhler, Jonas, Daniel Bernau, and Florian Kerschbaum. "Privacy-preserving outlier detection for data streams." *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, Cham, 2017.
- [7] M.E Andrés, N.E. Bordenabe, K.Chatzikokolakis, and C.Palamidessi. *Geo-indistinguishability: Differential privacy for location-based systems*. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013
- [8] F. McSherry, and K.Talwar. *Mechanism design via differential privacy*. Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on. IEEE, 2007
- [9] Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J., Vilhuber, L.: *Privacy: Theory meets practice on the map*. In: Proceedings of the International Conference on Data Engineering (ICDE) 2008
- [10] Canteaut, Anne, et al. "Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression." *Journal of Cryptology* 31.3 (2018): 885-916.
- [11] Fan, Junfeng, and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." *IACR Cryptology ePrint Archive* 2012 (2012): 144.
- [12] Rachel Player. "Parameter selection in lattice-based cryptography" PhD Thesis (2017)
- [13] Guillaume Bonnoron. "A journey towards practical Fully Homomorphic Encryption" PhD Thesis (2018)
- [14] Albrecht, Martin R., et al. "Revisiting the expected cost of solving uSVP and applications to LWE." *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Cham, 2017.
- [15] Elena Kirshanova. "Complexity of the learning with errors problem and memory-efficient lattice sieving" PhD Thesis, 2016.
- [16] Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." *Journal of the ACM (JACM)* 56.6 (2009): 34.
- [17] Albrecht, Martin R., Rachel Player, and Sam Scott. "On the concrete hardness of learning with errors." *Journal of Mathematical Cryptology* 9.3 (2015): 169-203.

- [18] Mina Sheikhalishahi, Andrea Saracino, Mohamed Mejri, Nadia Tawbi, Fabio Martinelli, “*Fast and Effective Clustering of Spam Emails Based on Structural Similarity*”. FPS 2015: 195-211
- [19] Mina Sheikhalishahi, Andrea Saracino, Mohamed Mejri, Nadia Tawbi, Fabio Martinelli, “*Digital Waste Sorting: A Goal-Based, Self-Learning Approach to Label Spam Email Campaigns*”. STM 2015: 3-19
- [20] Jaehong Park and Ravi Sandhu. "The UCONABC usage control model.". ACM Trans. Inf. Syst. Secur. 7, 1 (2004), 128-174.

Appendix 1. DSA parent ontology for DSA Editor

The upper vocabulary file is inherited by all the C3ISP vocabularies and provides the basic structure for the CNL language.



upper_vocabulary.owl