Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare

Project Acronym: **ASCLEPIOS**
Project Contract Number: **826093**

Programme**: Health, demographic change and wellbeing**
Call: **Trusted digital solutions and Cybersecurity in Health and Care to protect privacy/data/infrastructures**
Call Identifier: **H2020-SC1-FA-DTS-2018-2020**
Focus Area: **Boosting the effectiveness of the Security Union**
Topic**: Toolkit for assessing and reducing cyber risks in hospitals and care centres**
Topic Identifier: **H2020-SC1-U-TDS-02-2018**
Funding Scheme: **Research and Innovation Action**
Start date of project: 01/12/2018 Duration: 36 months

Deliverable:

D3.2 ASCLEPIOS Models Editor and Interpretation Mechanism

Due date of deliverable: 31/3/2020 Actual submission date: 07/04/2020

WPL: ICCS

Dissemination Level: Public

Version: Final

ASCLEPIOS is supported by the H2020 Programme under contract no. 826093

Table of Contents

List of Figures and Tables

Figures

Tables

Status, Change History and Glossary

Status:	Name:	Date:	Signature:
Draft:	Yiannis Verginadis	24/3/2020	
Reviewed:	Krefting Dagmar	6/4/2020	
Approved:	Tamas Kiss	7/4/2020	

Table 1: Status Change History

Version	Date	Pages	Author(s)	Modification
v1.0	26/1/2020	14	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra	Initial draft (with Overview Diagrams of Context Element, Details of context elements (tables))
v1.1	26/2/2020	40	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra	ABAC Policy Editor section added
v1.2	15/03/2020	65	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra	ABE Policy Editor section added
v2.0	24/03/2020	68	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra	Pre-final version without security audit report
v2.1	24/03/2020	65	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra, Panayiotis Charalambous, Christiaan Hillen	Pre-final version with Secura’s security audit report
Final	07/04/2020	74	Yiannis Verginadis, Dimitris Apostolou, Ioannis Patiniotakis, Gregoris Mentzas, Jenny Psarra, Panayiotis Charalambous, Christiaan Hillen	Final version

Table 2: Deliverable Change History

ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
AMPLE	ASCLEPIOS Models and PoLicies Editors
AJAX	Asynchronous JavaScript and XML
CASM	Context-Aware Security Model
CSS	Cascading Style Sheets (CSS)
DOM	Document Object Model
EHR	Electronic Health Record
JAR	Java archive file
JPA	Java Persistence API
JSON	JavaScript Object Notation
JSP	Java Server Pages
RDF	Resource Definition Framework
REST	Representational state transfer
SPARQL	SPARQL Protocol and RDF Query Language
TTL	Turtle syntax
UML	Unified Modelling Language
URI	Unified Resource Identifier
XACML	eXtensible Access Control Markup Language

Table 3: Glossary

Executive Summary

Despite the recent adoption growth of cloud computing and the value of hosting sensitive information systems in virtualised resources, a lot of users seem to be reluctant to store their personal data in the cloud or adopt sensitive systems hosted in the cloud. This is even more evident when the systems in question refer to the healthcare domain. Our goal is to enhance the access control mechanisms that can be used in the healthcare domain and raise the security awareness. In this way, the users' trust will be extended and their unwillingness to use cloud-based applications for their health data will be hopefully diminished. In this work, we report on the development of all the appropriate mechanisms for updating the ASCLEPIOS context-aware security model and devising the necessary context-aware access policies. These access control policies will be enforced as part of two different authorisation paradigms that will be employed in ASCLEPIOS in sequence for achieving even higher levels of security controls. These paradigms are the Attribute-based Access Control (ABAC) and the Attribute-based Encryption (ABE). This deliverable also reports on the editing functionalities for creating context-aware access policies (ABAC and ABE) and the interpretation capabilities for exporting these policies in the appropriate format to enable the access enforcement mechanisms.

1 Introduction

WP3 focuses on defining and evaluating contextual information (e.g., the identity of a user, its role, patterns of access, connection type etc.) and attributes that characterise sensitivity levels of data. This information is considered in ASCLEPIOS before granting any data access request. Based on such information the contextual model was developed in D3.1 [1] and building on it a number of enforcement rules can be created as the most elementary structural elements of policies. Indicatively, attributes that are organized in a hierarchical structure may include concepts related to: i) the device from which there is an access attempt, ii) the actor that tries to access the data (e.g. location, IP, role in the healthcare use case, etc.) and iii) historic data that reveal patterns of access (e.g. frequency, usual dates or hours of access, usual duration of access, previously accessed data, etc.). Such concepts along with a number of properties that interrelate them, serve as background knowledge for the ASCLEPIOS access control policies. These access control policies are then enforced as part of two different authorisations paradigms that are employed in ASCLEPIOS in sequence for achieving even higher levels of security controls. These paradigms are the Attribute-based Access Control (ABAC) and the Attribute-based Encryption (ABE).
This deliverable reports on the development of all the appropriate mechanisms for updating the contextual model and devising the context-aware access policies, i.e., editing functionalities for creating context-aware access policies (ABAC and ABE) and the interpretation mechanism that will export these policies in the appropriate format for enabling the access enforcement mechanisms. The purpose of this document is two-fold. First is describes in a fine-grained way each related mechanism's purpose, value and overall contribution to the ASCLEPIOS framework, and second it reveals and comments on the details of the mechanisms' implementation.

I.1. Objectives

The primary objectives of this deliverable are to:

1. Provide the necessary editor for improving and extending the ASCLEPIOS context-aware security model;
2. Develop the appropriate editing functionalities for allowing DevOps of cloud-based eHealth systems to declaratively create the minimum amount of rule-set that needs to be enforced for security purposes and organise it across ABAC and ABE policies;
3. Dynamically interpret these annotations into formats that enable the ABAC and ABE authorisation mechanisms.

Following these objectives, the policies access, decision and enforcement components will be developed in terms of WP3.

I.2. Relationship to ASCLEPIOS Deliverables

This deliverable documents the editors for extending the ASCLEPIOS context-aware security model and defining context-aware access control policies. It utilises the modelling primitives defined and described in D3.1 [1] which constitutes the background knowledge for establishing authorization control over accessing EHR, based on ABAC and ABE paradigms. Moreover, this deliverable described the interpretation mechanism which will feed into the context-aware ABAC enforcement mechanism (to be reported in D3.3) and the ABE service (reported in D2.2 [2]).

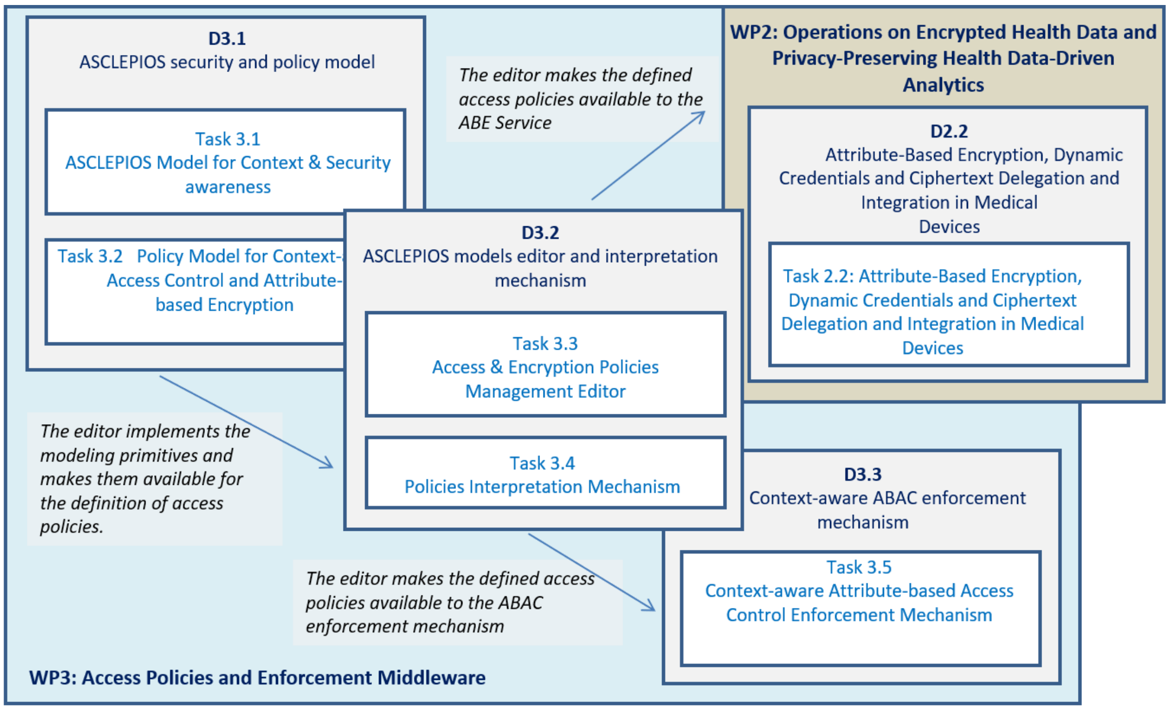

D3.1
ASCLEPIOS security and policy model
Task 3.1
ASCLEPIOS Model for Context & Security awareness
Task 3.2 Policy Model for Context-Access Control and Attribute-based Encryption
The editor makes the defined access policies available to the ABE Service
WP2: Operations on Encrypted Health Data and Privacy-Preserving Health Data-Driven Analytics
D2.2
Attribute-Based Encryption, Dynamic Credentials and Ciphertext Delegation and Integration in Medical Devices
Task 2.2: Attribute-Based Encryption, Dynamic Credentials and Ciphertext Delegation and Integration in Medical Devices
D3.2
ASCLEPIOS models editor and interpretation mechanism
Task 3.3
Access & Encryption Policies Management Editor
Task 3.4
Policies Interpretation Mechanism
The editor implements the modeling primitives and makes them available for the definition of access policies.
D3.3
Context-aware ABAC enforcement mechanism
Task 3.5
Context-aware Attribute-based Access Control Enforcement Mechanism
The editor makes the defined access policies available to the ABAC enforcement mechanism
WP3: Access Policies and Enforcement Middleware

Figure 1: Relation of D3.2 with other WP3 and WP2 deliverables

The model presented herein will constitute the necessary background knowledge layer for enabling the ABAC and ABE paradigms in ASCLEPIOS. Specifically, policies will be determined through a number of attributes that reveal valuable security-related details of the entity that is requesting access to sensitive data, the data itself and its ambient environment. Hence, the model will serve the development of the Data Access Policies Interpretation and Enforcement mechanism (WP3) as well as the ABE service (WP2).

I.3. Organization

Chapter 1 is this introduction that describes the development of the editors and the interpretation mechanism. Chapter 2 outlines the overall conceptual and physical architecture of the editor and the interpretation mechanism as well as the technologies and tools used for their implementation. Chapter 3 describes the Context-Aware Security Model editor, chapter 4 the ABAC policies editor and chapter 5 the ABE editor. Finally, the information presented in this document is summarized in the final chapter (chapter 6).

2 Overall Approach and Architecture

In the present chapter the *ASCLEPIOS Models and PoLicies Editors*, or *AMPLE for simplicity*, are introduced and detailed.

The purpose of AMPLE is to provide all necessary design-time tools for creating, maintaining and verifying access control policies of ASCLEPIOS platform. Since ASCLEPIOS considers two approaches for access control, namely Attribute-Based Access Control (ABAC) and Attribute-Based Encryption (ABE), AMPLE provides two editors for developing the corresponding ABAC and ABE policies. It will furthermore provide a Policy Validation module, where policy developers can define rules for checking policy correctness, completeness or for security awareness, but this editor will be part of D3.3 and is not covered by this deliverable.
AMPLE moreover provides a configurable, common vocabulary for application-related attributes, which can be further tailored to each application's needs. This is crucial since both access control methods (ABAC and ABE) rely on the use of attributes. Therefore, it is important to use attributes in a semantically coherent manner. This common vocabulary is called *Context-Aware Security Model (CASM)* and is stored in AMPLE's internal repository. It furthermore offers an editor for displaying and modifying CASM. More information on CASM can be found in deliverable D3.1 [1].
In user experience terms, AMPLE aspires to offer a unified environment of graphical tools for creating, maintaining and validating ASCLEPIOS access control policies. Specifically, it has been implemented as a web-application, hence allowing its easy usage. To this end various modern Web 2.0 technologies have been used.

I.1. Conceptual Architecture

In this section the conceptual architecture of AMPLE is presented. AMPLE is a unified environment encompassing several tools; three editors, Models Store (repository), Policies interpreter and a Policies Validation module. Figure 2 provides a visual representation of the conceptual architecture where the main tools are also depicted. Note that the Policies Validation is mentioned to give the full picture. It will be described in a subsequent deliverable D3.3.

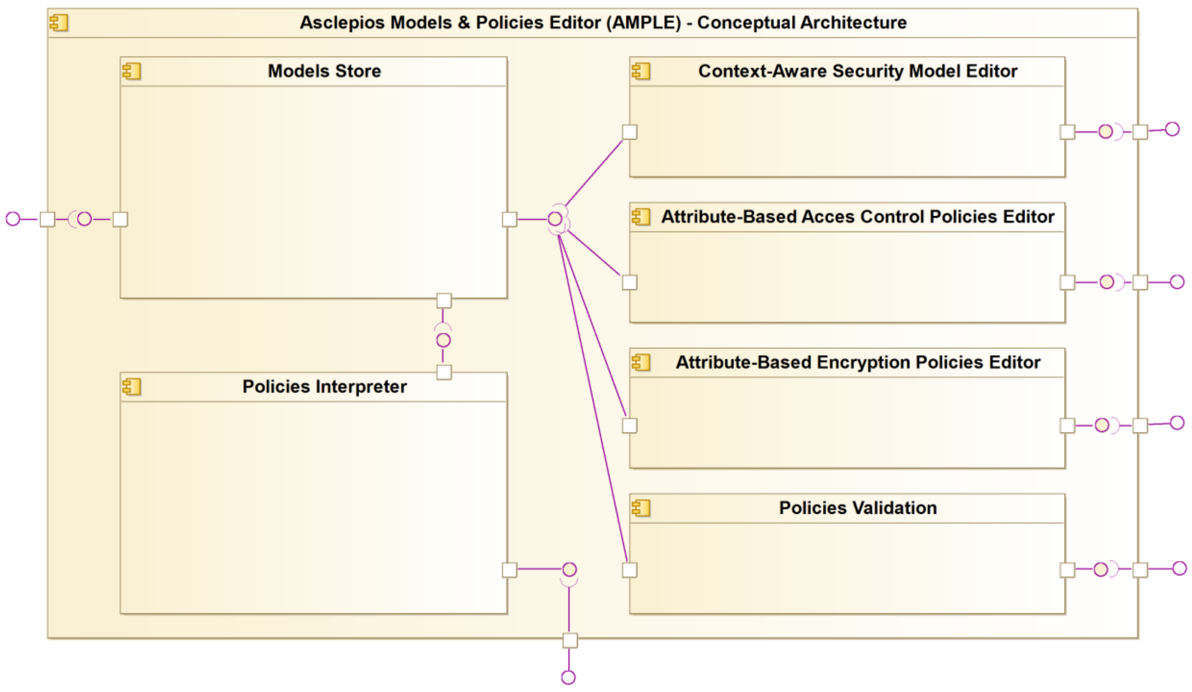

Asclepios Models & Policies Editor (AMPLE) - Conceptual Architecture
Models Store
Context-Aware Security Model Editor
Attribute-Based Acces Control Policies Editor
Attribute-Based Encryption Policies Editor
Policies Interpreter
Policies Validation

Figure 2 – AMPLE conceptual architecture

Following, the main components of the architecture are described.

- ***Context-Aware Security Model Editor.*** Provides the means for creating and maintaining the Context-Aware Security Model (CASM). CASM paves the path for defining ABAC and ABE policies by using a common vocabulary. CASM Editor offers both a web-based, graphical interface for representing and modifying CASM, as well as the necessary model implementations. CASM is stored in Models Store, thus is made available to other AMPLE tools. More information on CASM can be found in deliverable D3.1 [1].
- ***Attribute-Based Access Control Policies Editor.*** Provides a graphical interface for defining and modifying Attributed-Based Access Control (ABAC) policies. ABAC policies are stored in Models Store, thus are made available to other AMPLE tools.
- ***Attribute-Based Encryption Policies Editor.*** Provides graphical interface for defining and modifying Attributed-Based Encryption (ABE) policies. ABE policies are stored in Models Store, thus are made available to other AMPLE tools. Through the ABE Policies Editor interface the selected ABE policies are provided as input to the ABE Service (D2.2 [2]).
- ***Policies Validation.*** Provides the means to check ABAC and ABE policies against a given set of rules. These rules might check the conformity, validity, correctness or completeness of policies. This tool offers a graphical interface for defining the rules as well as applying them to selected policies. Policies Validation module will be developed and reported as part of the upcoming deliverable D3.3.
- ***Models Store.*** It is a repository for persisting all kinds of models handled in AMPLE; i.e. Context-Aware Security Model, ABAC policies, ABE policies and Policy Validation rules. It internally uses a Resource Description Framework (RDF[1]) triple store for storing the models as well as a layer for serializing model objects (i.e. core elements of the model) to RDF and vice versa.
- ***Policies Interpreter.*** This tool translates ABAC policies captured as RDF graphs in Models Store into proper eXtensible Access Control Markup Language (XACML) documents that can be used by any XACML-capable access control engine

[1] https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

(following the ABAC paradigm), or in the appropriate format that expects the ABE service of ASCLEPIOS (D2.2 [2]).

I.2. Techical Architecture

AMPLE implementation realized the conceptual architecture presented before, with regard to the provided functionality and overall approach. the actual (technical) structure, components and interconnections of AMPLE parts are explained in physical architecture, presented next.

In technical terms, AMPLE is a web-based application, encompassing both a server-side part as well as a client-side part that offers the graphical user interface (as depicted in Figure 3).

Figure 3 Figure 3 also depicts the two ASCLEPIOS Policy Enforcement Mechanisms that are supplied with the policies created with AMPLE, as well as the possibility for third-party Representational State Transfer (REST) clients to interact with AMPLE in order to reuse or modify AMPLE models (i.e. CASM, ABAC and ABE policies and Policy Validation rules). Next, we provide additional details for each of components of the AMPLE physical architecture.

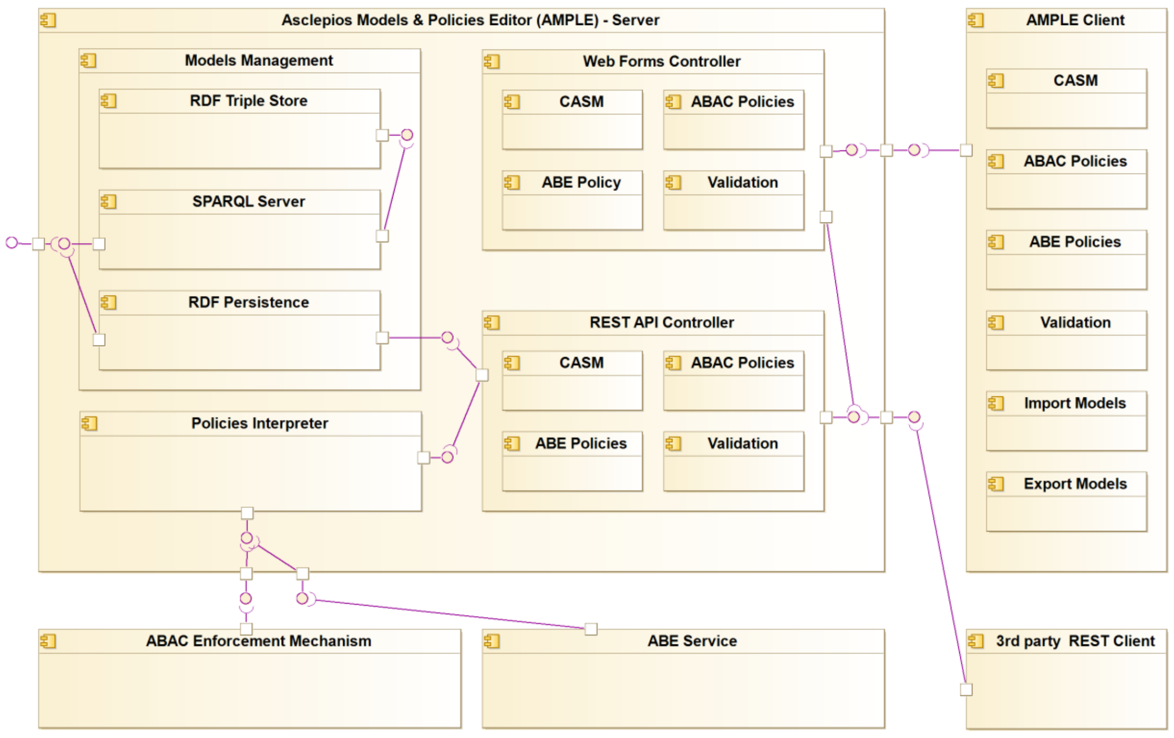


Figure 3 – AMPLE technical architecture

AMPLE Server

AMPLE Server is the core component of the AMPLE physical architecture. It interacts with the AMPLE client, the Policy Enforcement Mechanism or third-party REST clients. It internally comprises of the following parts.

- ***Web Forms controllers*** are responsible for the interaction with the AMPLE client forms, specifically for providing the requested information and collecting the submitted models. Web Forms controllers rely on REST API controllers both for retrieving the requested information and for saving the submitted models. Essentially, they act as a translation layer between REST API and AMPLE client forms, by

turning AMPLE client requests into proper REST API requests and vice versa.
There are four Web Form controllers:

- ***CASM Web Form controller,*** which is responsible for the CASM related interactions with AMPLE client.
- ***ABAC Policies Web Form controller,*** which is responsible for the ABAC Policies related interactions with AMPLE client, and also for triggering the interpretation of ABAC policies to XACML and sending them to Policies Enforcement Mechanism.
- ***ABE Policies Web Form controller,*** which is responsible for the ABE Policies related interactions with AMPLE client.
- ***Policies Validation Web Form controller,*** which is responsible for the Policies Validation related interactions with AMPLE client as well as for initiating the policy validation process.

- ***REST API controllers*** accept REST requests for retrieving model related information (or whole models) and also for storing models, using a specific JavaScript Object Notation (JSON) format. REST API controllers are used by Web Forms controllers (which act as REST clients) but third-party REST clients can also interact with REST API controllers, as long as they are capable to handle the JSON format used and as long as the AMPLE server is configured to accept REST API requests from external clients.
  There are four REST API controllers:
  - ***CASM REST API controller,*** which is responsible for storing and retrieving CASM-related concepts like attributes and attribute properties.
  - ***ABAC Policies REST API controller,*** which is responsible for storing and retrieving ABAC Policies and ABAC Policy Rules used to build ABAC Policies.
  - ***ABE Policies REST API controller,*** which is responsible for storing and retrieving ABE Policies.
  - ***Policies Validation REST API controller,*** which is responsible for storing and retrieving Policies Validation rules.

- ***Models Management*** is responsible for storing and retrieving CASM attributes, ABAC policies and rules, ABE policies and Policies Validation rules. Its internal persistence mechanism is based on an RDF triple store, thus all information is stored and retrieved as RDF triples forming RDF graphs. For this reason, the Models Management component also encompasses a layer that serializes model objects into RDF graphs (representing an object) and vice versa.
  The Models Management component comprises of three subcomponents:
  - ***RDF Triple store,*** which is responsible for persisting and retrieving RDF graphs describing model objects.
  - ***SPARQL server,*** which is responsible for accepting and carrying out queries for retrieving and modifying the persisted RDF graphs in RDF triple store. The query language used is SPARQL[2].
  - ***RDF Persistence,*** which is responsible for converting model objects into SPARQL queries that are submitted to SPARQL server and persist the state of the model objects into RDF triple store. Reversely, RDF Persistence can build SPARQL queries for retrieving the persisted state of a model object from RDF Triple store and convert the retrieved RDF triples into the corresponding model object, which can subsequently be used in REST API controllers or other AMPLE server components.

- ***Policies Interpreter*** is responsible for: i) retrieving the specified ABAC policies and translating them into XACML documents, which are then submitted to the ABAC Policies Enforcement Mechanism and ii) retrieving the specified ABE policies and

[2] https://www.w3.org/TR/sparql11-query/

translating them into an appropriate text format (described in section 5.5) , which are then submitted to the ABE service (D2.2 [2]).

AMPLE Client

It is the client-side part of AMPLE. It consists of a set of dynamic web pages that provide the graphical user interface of each AMPLE tool, as well as some common graphical elements like the menu. The web pages are dynamically generated at server-side, using Java Server Pages (JSP), and then are rendered in the user's browser. In response to user actions, web pages can contact the corresponding Web Forms controllers in order to send or retrieve the needed model information.

Policy Enforcement Mechanism

Policy Enforcement Mechanism is a core component of the overall ASCLEPIOS architecture. It is responsible for enforcing the ABAC access control policies, created with AMPLE (and possibly other tools too). More information on Policy Enforcement Mechanism will be provided in deliverable D3.3.

ABE Service

The Attribute-Based Encryption (ABE) Service is a core component of the ASCLEPIOS architecture. It is responsible for protecting resources by encrypting them using ABE Policies created with AMPLE, as well as allowing access to them based on user attributes and the ABE Policy used.

Third-party REST API clients

AMPLE provides a REST API for retrieving and manipulating various models (CASM, ABAC and ABE policies, and policy validation rules). If configured accordingly, it can accept REST calls from external (third-party) tools both for retrieving and for submitting models stored in Models Store. Third-party clients must be REST-capable and aware of the message formats, endpoints and REST verbs used. Furthermore, if they need to process the model information (received through REST API) they will also need to be aware of the model semantics used in AMPLE and ASCLEPIOS.

I.3. Implementation

AMPLE server has been implemented using the Java™ programming language, version 8. It has been developed as a Spring-boot web application, for easier dependency management and customization of component properties. It is built and bundled, using the well-known Maven system, into a single fat JAR, containing all necessary dependencies. It is also bundled (during its build) as a Docker container. From a design perspective it follows the Model-View-Controller paradigm.

AMPLE server also encompasses an instance of Apache Jena Fuseki[3] SPARQL server, backed by Apache Jena TDB[4] as its RDF Triple store[5]. These form the foundation of AMPLE Models Store. Moreover, a custom library for converting programming objects representing models (or model parts) into SPARQL queries has been developed. This library provides a Java Persistence API (JPA)-like approach of storing, modifying and retrieving model objects to/from RDF Triple store via Fuseki SPARQL server.

3 https://jena.apache.org/documentation/fuseki2/index.html
4 https://jena.apache.org/documentation/tdb/index.html
5 https://jena.apache.org/

AMPLE Client uses Web 2.0 technologies like Document Object Model (DOM), Asynchronous JavaScript and XML (AJAX), JavaScript, Cascading Style Sheets (CSS), and JSON. It also makes use of the well-known jQuery[6] library as well as several plugins, and Bootstrap CSS framework[7], for creating its graphical user interface.

AMPLE source code is available at GitLab.com : https://gitlab.com/asclepios/ample-editor

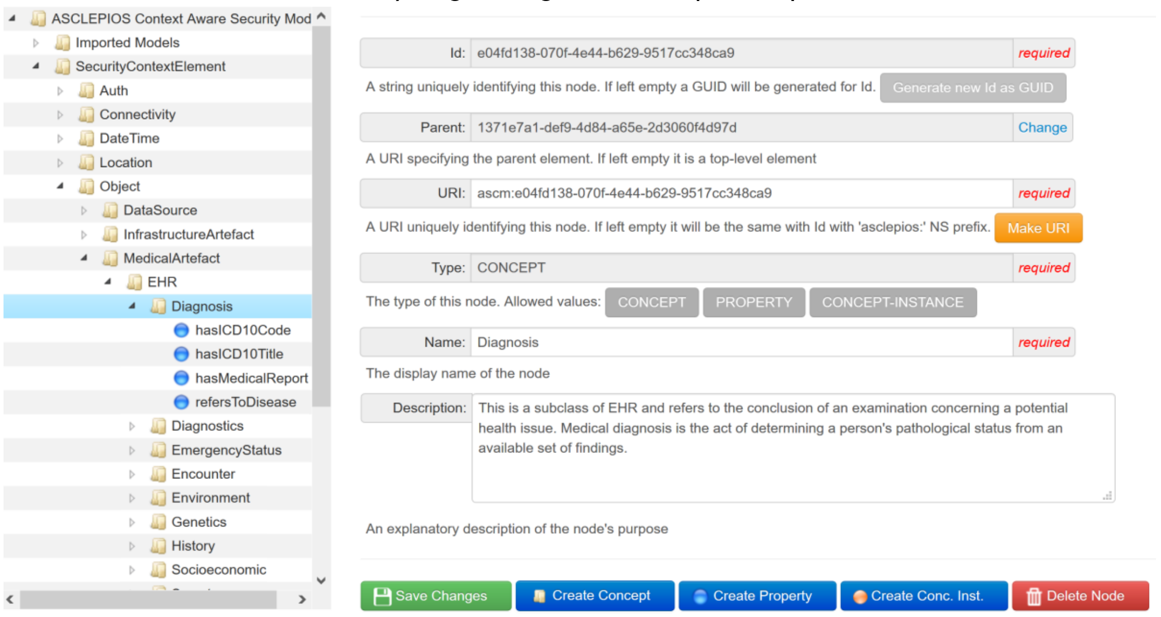
In the following chapters the three AMPLE editors will be further detailed.

[6] https://jquery.com/
[7] https://getbootstrap.com/

3 Context–Aware Security Model (CASM) Editor

In this chapter the Context–Aware Security Model (CASM) Editor is presented and detailed. However, it would be beneficial to recap what CASM is and how it is structured, before getting into the editor's details.

As it has been previously noted, CASM provides a configurable, common vocabulary of attributes, which ensures the semantically coherent usage of them in other tools. CASM is a hierarchical (tree-like) taxonomy of attributes (referred as *Concepts*), attribute properties, and attribute instance values (or just instances) when they are known beforehand. An attribute is titled with a name, and uniquely identified by a Universal Resource Identifier (URI). It furthermore has an (optional) description which describes its exact semantics. An attribute can have sub-attributes, which are specializations of the parent attribute's meaning. It can also have attribute instances, when they are a priori known, as well as properties that can relate attribute (as a concept) to other attributes or common data types (like numbers, date/time, literals, and Booleans). Figure 4 gives a sample snapshot of CASM.



ASCLEPIOS Context Aware Security Mod
- Imported Models
- SecurityContextElement
  - Auth
  - Connectivity
  - DateTime
  - Location
  - Object
    - DataSource
    - InfrastructureArtefact
    - MedicalArtefact
      - EHR
        - Diagnosis
          - hasICD10Code
          - hasICD10Title
          - hasMedicalReport
          - refersToDisease
        - Diagnostics
        - EmergencyStatus
        - Encounter
        - Environment
        - Genetics
        - History
        - Socioeconomic

Id: e04fd138-070f-4e44-b629-9517cc348ca9 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id. Generate new Id as GUID

Parent: 1371e7a1-def9-4d84-a65e-2d3060f4d97d Change
A URI specifying the parent element. If left empty it is a top-level element

URI: ascm:e04fd138-070f-4e44-b629-9517cc348ca9 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix. Make URI

Type: CONCEPT *required*
The type of this node. Allowed values: CONCEPT PROPERTY CONCEPT-INSTANCE

Name: Diagnosis *required*
The display name of the node

Description: This is a subclass of EHR and refers to the conclusion of an examination concerning a potential health issue. Medical diagnosis is the act of determining a person's pathological status from an available set of findings.
An explanatory description of the node's purpose

Save Changes | Create Concept | Create Property | Create Conc. Inst. | Delete Node

Figure 4 – Sample excerpt of the default CASM

In the following sections the CASM editor is presented in detail.

I.1. Usage Scenarios

The main goal of CASM Editor is to manage the CASM elements and structure. This goal can be broken down to a series of specific capabilities that must be offered to the user for creating, updating, deleting and retrieving the CASM attributes, values and properties, as well as for importing and exporting CASM to a file. A full list of CASM Editor capabilities (and related usage scenarios) is given next.
The term *Elements* in the following list and diagrams is a collective reference to attributes, properties and instance values.

- Display CASM hierarchy (tree)
- Retrieve an Element's details
- Create a new Element in CASM
- Modify or Delete an existing Element in CASM
- Initialize AMPLE with default CASM (shipped with AMPLE)
- Import CASM from a file into AMPLE

- Export CASM from AMPLE to a file

The CASM retrieval and editing capabilities are depicted in Figure 5 whereas the import/export capabilities are depicted in Figure 6.

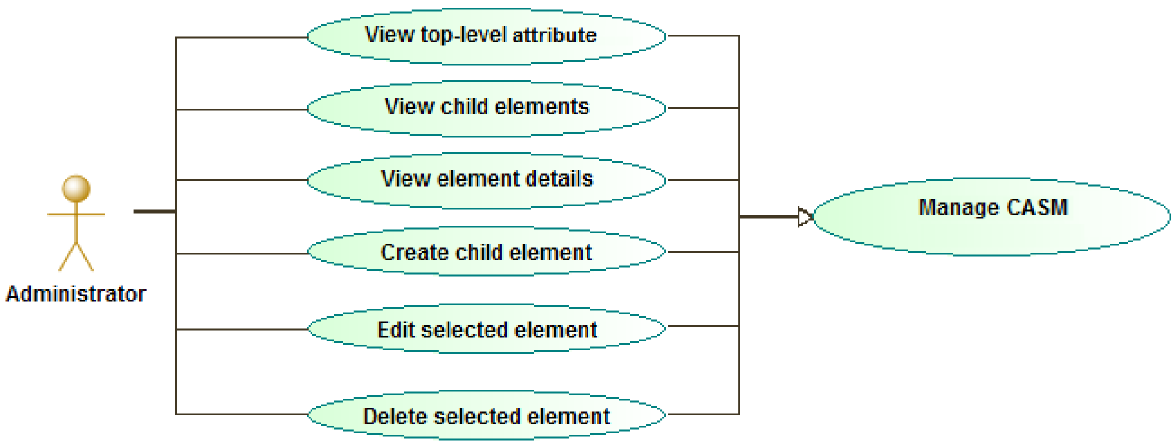


Figure 5 – CASM management use case

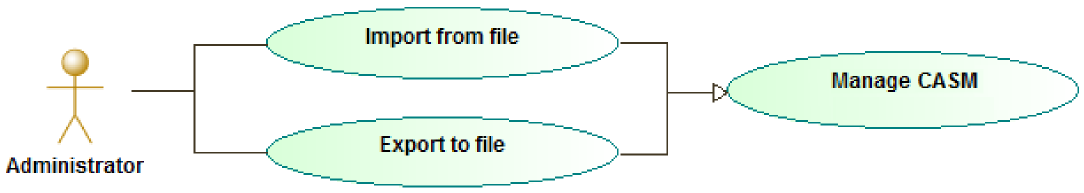


Figure 6 – CASM Import/Export use case

I.2. Walkthrough

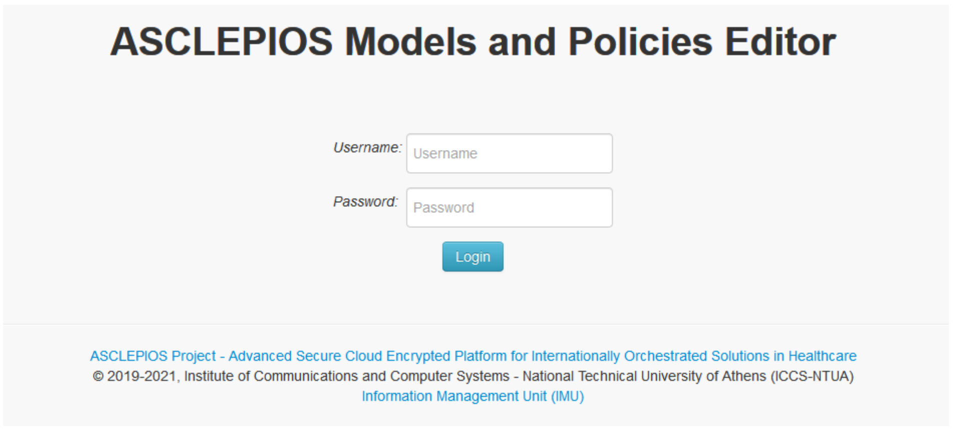
In this section the graphical user interface of the CASM Editor is briefly presented along with a short walkthrough of its operation. The editor can be accessed through the AMPLE web page, therefore an overview of common AMPLE pages is given first.

Login to AMPLE

AMPLE can be accessed using a modern web browser and typing its web address, i.e. the server where AMPLE has been installed. Typically, it is something like:

`https://<SERVER>:9090/`

The user will land to the AMPLE Login page where valid user credentials must be entered in order to be authenticated (see Figure 7).



ASCLEPIOS Models and Policies Editor

Username: Username

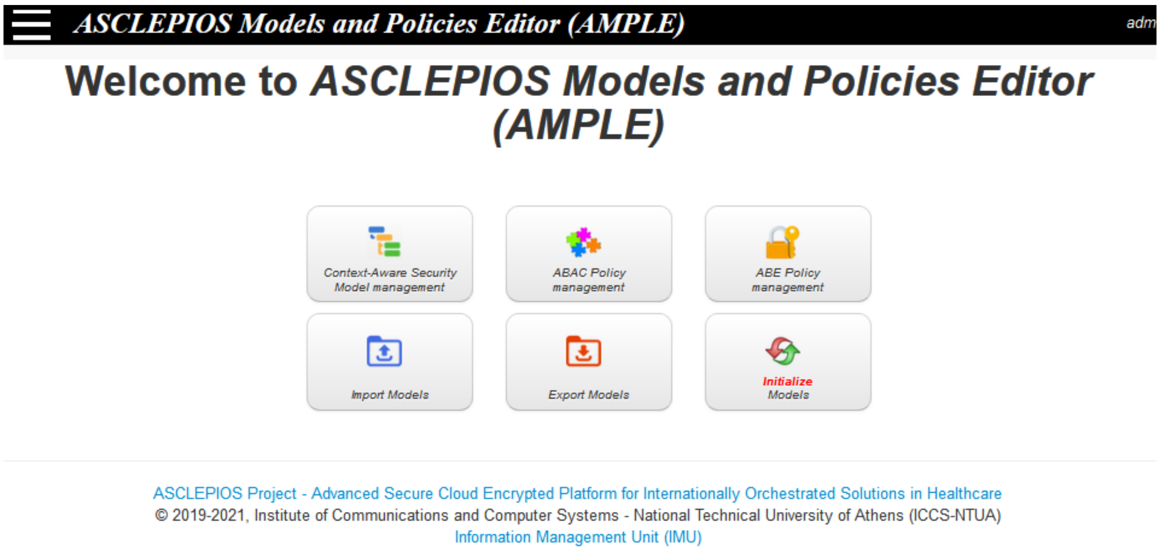
Password: Password

Login

ASCLEPIOS Project - Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare
© 2019-2021, Institute of Communications and Computer Systems - National Technical University of Athens (ICCS-NTUA)
Information Management Unit (IMU)

Figure 7 – AMPLE login page

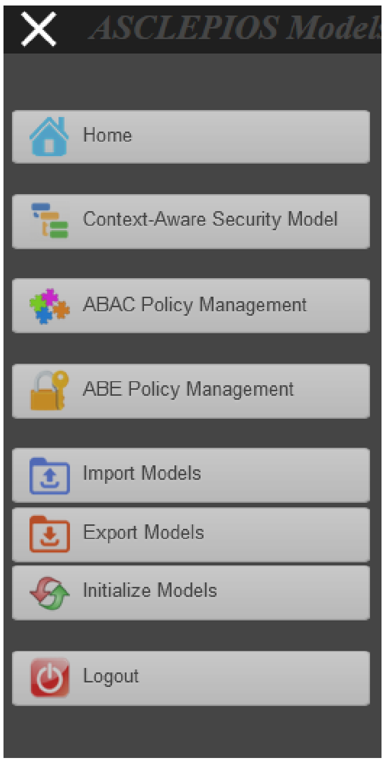
On successful authentication the user is directed to AMPLE Welcome page (see Figure 8). Welcome page offers a number of buttons that will launch the corresponding operations or tools of AMPLE. The same functionality can also be accessed from the sliding menu that appears when clicking on ☰ sign on the upper left part of the page. Figure 9 depicts AMPLE menu along with short explanation of each button's operation.



Welcome to *ASCLEPIOS Models and Policies Editor (AMPLE)*

ASCLEPIOS Project - Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare
© 2019-2021, Institute of Communications and Computer Systems - National Technical University of Athens (ICCS-NTUA)
Information Management Unit (IMU)

Figure 8 – AMPLE welcome page



Home: Returns to the Welcome page

Context-Aware Security Model: Opens the CASM Editor

ABAC Policy Management: Opens the ABAC Policy Editor

ABE Policy Management: Opens the ABE Policy Editor

Import Models: Opens the Import page

Export Models: Starts downloading Model Store contents

Initialize Models: Initializes Models Store with CASM shipped with AMPLE

Logout: Disconnects user from AMPLE

Figure 9 – AMPLE menu

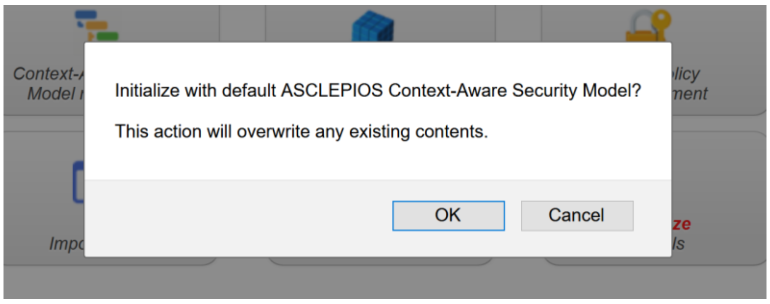
Initialize CASM

The first time (after installation) that AMPLE is launched, Models Store will be empty. The user has three options with regard to CASM initialization:

1. Create a new CASM from scratch
2. Import CASM from a file exported by another AMPLE instance
3. Initialize Models Store using the CASM shipped with AMPLE

It is expected that the third option will be used when defining a new application within the ASCLEPIOS framework and therefore needing to configure CASM to its specific needs. The second option can be used either when a new application shares the same concepts and attributes with another (pre-existing) one, or when migrating or replicating AMPLE to other servers. In the latter case an export is required from the original AMPLE instance, which can subsequently used to initialize new AMPLE instances.

In order to initialize Models Store using the shipped CASM, the user needs to press the button named "Initialize Models" and confirm the operation when asked (Figure 10).


Initialize with default ASCLEPIOS Context-Aware Security Model?

This action will overwrite any existing contents.

Figure 10 – Confirm CASM initialization

Therefore, the user can use the default CASM model by clicking on the "Initialize Models" option, or she can import a custom CASM model (e.g. from another application). In the latter case she must use the Import page to import custom CASM.

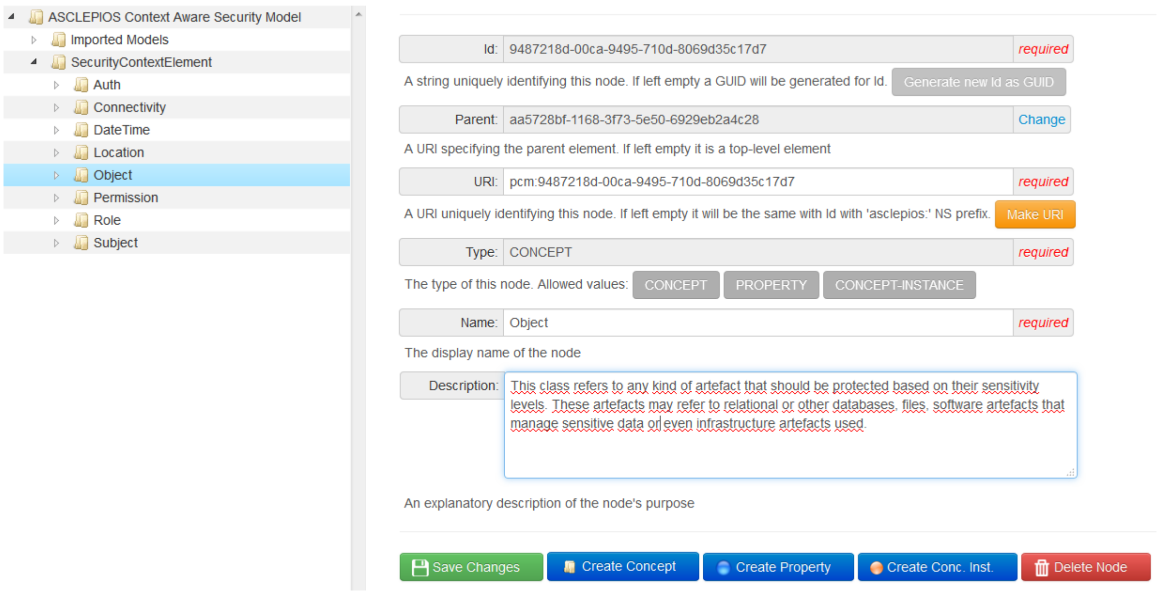
Note: *"Initialize Models"* must be used with caution because it replaces any previous contents of Models Store.

CASM Editor

CASM Editor opens by pressing the button named *"Context-Aware Security Model"* in the Welcome screen or the respective button in menu. Figure 11 gives an overview of CASM Editor.

The page is vertically divided in two notable regions. The left-hand part of the page contains a rendering of CASM in a tree-like fashion. The user can click on the arrow heads on the left side of each element in order to expand it and view its child elements (if any). Clicking on an element loads its details into the details form in the right-hand side of the page. Right clicking on an element in the CASM tree will open the context menu which offers actions related to the selected element (Figure 12).

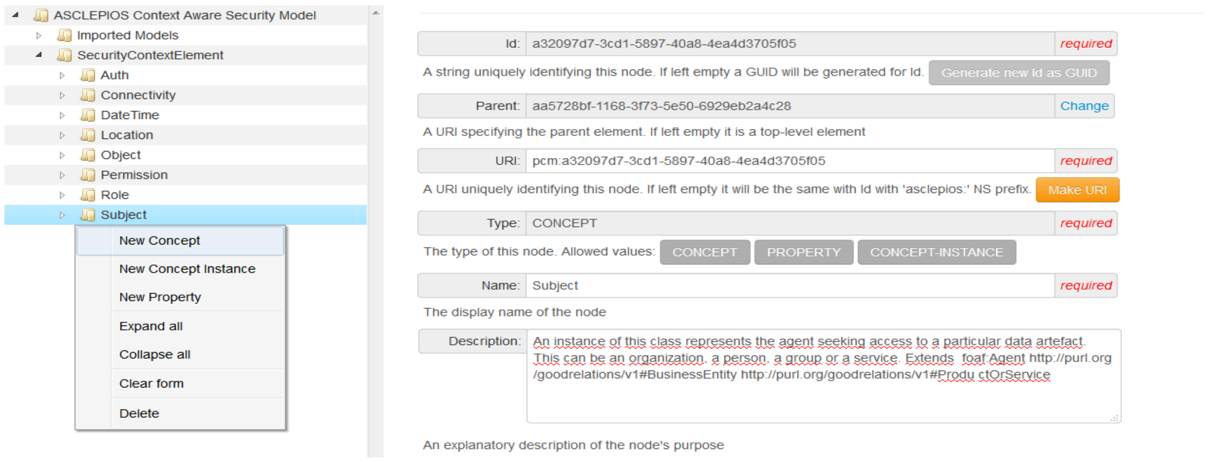
The details form (in the right-hand side of the page) encompasses fields that are common to all element types. Fields specific to particular element types are displayed only when an element of the corresponding type is selected. Under the details form, a row of buttons exists. These buttons can be used to save any changes made in details form, create new child elements or delete the currently displayed element in details form. Buttons get dimmed when the corresponding operation is not available in a particular case (e.g. when editing a property, creating child elements is not active).



- ASCLEPIOS Context Aware Security Model
  - Imported Models
  - SecurityContextElement
    - Auth
    - Connectivity
    - DateTime
    - Location
    - Object
    - Permission
    - Role
    - Subject

Id:	9487218d-00ca-9495-710d-8069d35c17d7	required
A string uniquely identifying this node. If left empty a GUID will be generated for Id.		Generate new Id as GUID
Parent:	aa5728bf-1168-3f73-5e50-6929eb2a4c28	Change
A URI specifying the parent element. If left empty it is a top-level element		
URI:	pcm:9487218d-00ca-9495-710d-8069d35c17d7	required
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.		Make URI
Type:	CONCEPT	required
The type of this node. Allowed values:	CONCEPT PROPERTY CONCEPT-INSTANCE	
Name:	Object	required
The display name of the node		
Description:	This class refers to any kind of artefact that should be protected based on their sensitivity levels. These artefacts may refer to relational or other databases, files, software artefacts that manage sensitive data or even infrastructure artefacts used	
An explanatory description of the node's purpose		

Figure 11 – CASM Editor page



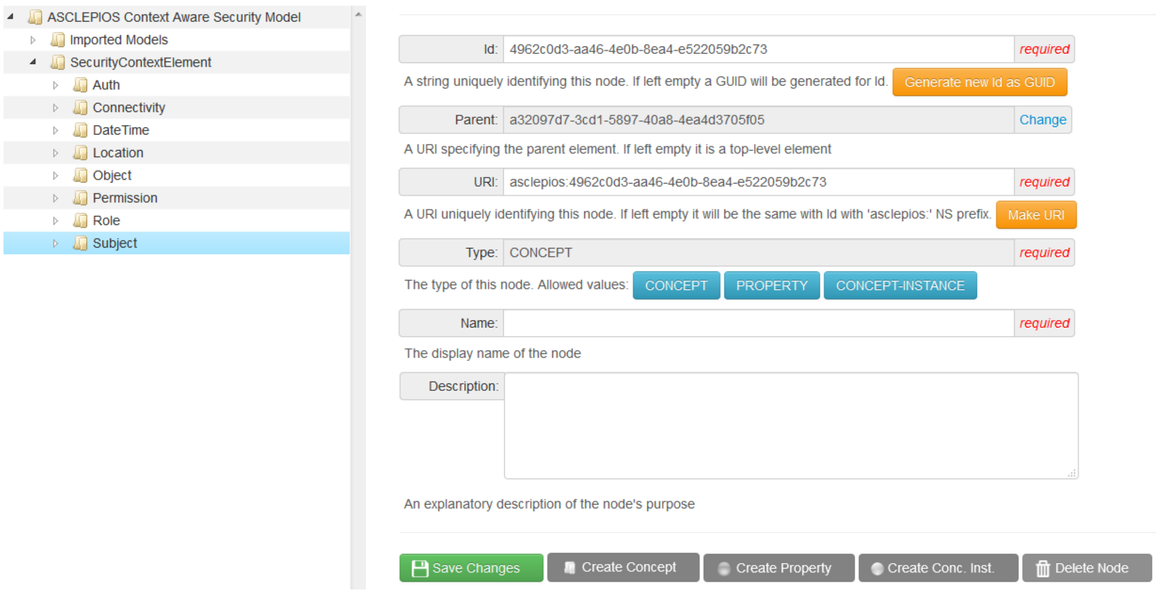
- ASCLEPIOS Context Aware Security Model
  - Imported Models
  - SecurityContextElement
    - Auth
    - Connectivity
    - DateTime
    - Location
    - Object
    - Permission
    - Role
    - Subject

Id:	a32097d7-3cd1-5897-40a8-4ea4d3705f05	required
A string uniquely identifying this node. If left empty a GUID will be generated for Id.		Generate new Id as GUID
Parent:	aa5728bf-1168-3f73-5e50-6929eb2a4c28	Change
A URI specifying the parent element. If left empty it is a top-level element		
URI:	pcm:a32097d7-3cd1-5897-40a8-4ea4d3705f05	required
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.		Make URI
Type:	CONCEPT	required
The type of this node. Allowed values:	CONCEPT PROPERTY CONCEPT-INSTANCE	
Name:	Subject	required
The display name of the node		
Description:	An instance of this class represents the agent seeking access to a particular data artefact. This can be an organization, a person, a group or a service. Extends foaf:Agent http://purl.org /goodrelations/v1#BusinessEntity http://purl.org/goodrelations/v1#Produ ctOrService	
An explanatory description of the node's purpose		

Figure 12 – Creation of a new attribute using context menu

Create new attribute

The user can create a new attribute (or other element) by first selecting the parent attribute in the CASM tree on the left, and then clicking on the corresponding option, either in the context menu (see Figure 12) or the buttons under the details form (see Figure 11).
Creating a new child element will clear details form, and prefill *Id*, *Parent* and *URI* fields with automatically created values (Figure 13). A user has the opportunity to modify these values before saving them (Figure 14). When ready the user can click on *Save* button (colored green) to submit information to the server for saving (Figure 15).



- ASCLEPIOS Context Aware Security Model
  - Imported Models
  - SecurityContextElement
    - Auth
    - Connectivity
    - DateTime
    - Location
    - Object
    - Permission
    - Role
    - Subject

Id: 4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

Parent: a32097d7-3cd1-5897-40a8-4ea4d3705f05

A URI specifying the parent element. If left empty it is a top-level element

URI: asclepios:4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: CONCEPT *required*

The type of this node. Allowed values: CONCEPT, PROPERTY, CONCEPT-INSTANCE

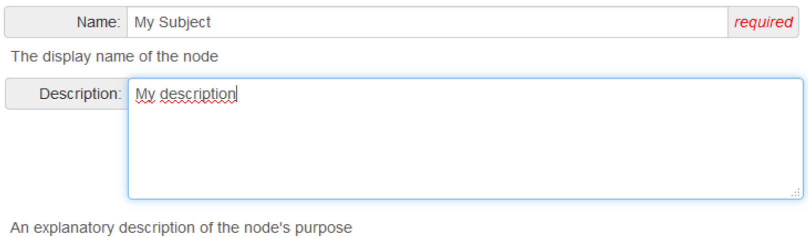
Name: *required*

The display name of the node

Description:

An explanatory description of the node's purpose

Figure 13 – New attribute form



Name: My Subject *required*

The display name of the node

Description: My description

An explanatory description of the node's purpose

Figure 14 – Filling in new attribute details

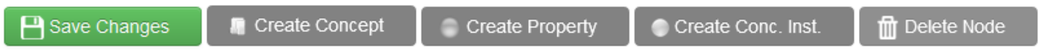
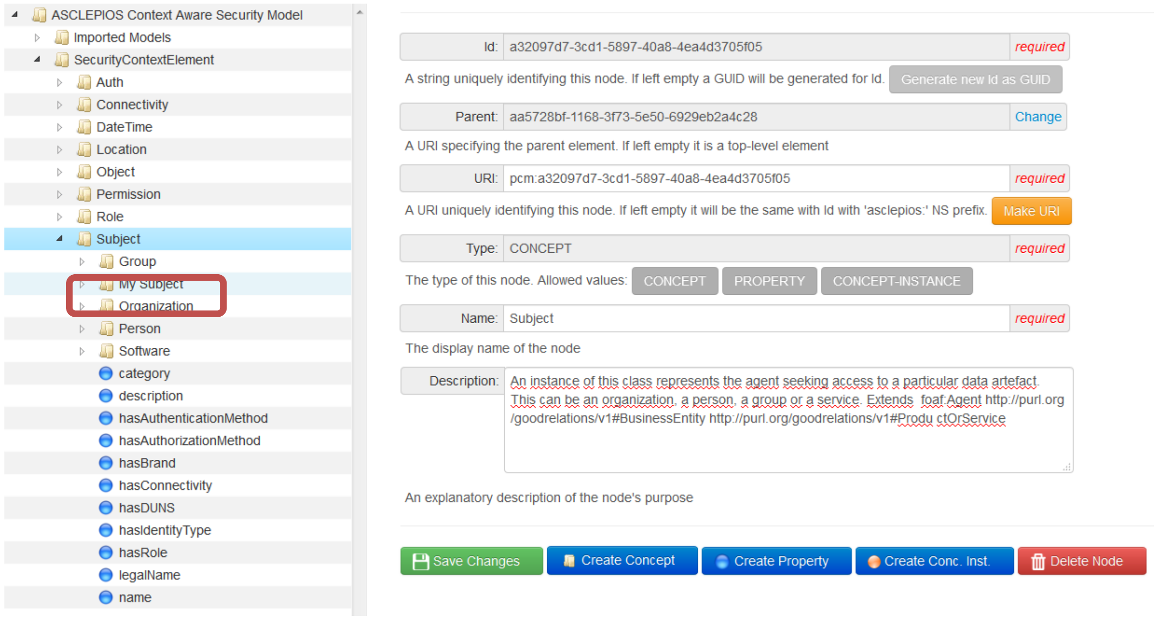


Figure 15 – Submit new attribute details

After saving changes, the CASM tree on the left-hand side of the page will refresh in order to reflect the changes (i.e. including the new attribute). If the parent attribute is not expanded, the user can click on its arrow head to expand its child elements (Figure 16).



- ASCLEPIOS Context Aware Security Model
  - Imported Models
  - SecurityContextElement
    - Auth
    - Connectivity
    - DateTime
    - Location
    - Object
    - Permission
    - Role
    - Subject
      - Group
      - My Subject
      - Organization
      - Person
      - Software
      - category
      - description
      - hasAuthenticationMethod
      - hasAuthorizationMethod
      - hasBrand
      - hasConnectivity
      - hasDUNS
      - hasIdentityType
      - hasRole
      - legalName
      - name

Id: a32097d7-3cd1-5897-40a8-4ea4d3705f05 *required*

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

Parent: aa5728bf-1168-3f73-5e50-6929eb2a4c28

A URI specifying the parent element. If left empty it is a top-level element

URI: pcm:a32097d7-3cd1-5897-40a8-4ea4d3705f05 *required*

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: CONCEPT *required*

The type of this node. Allowed values: CONCEPT, PROPERTY, CONCEPT-INSTANCE

Name: Subject *required*

The display name of the node

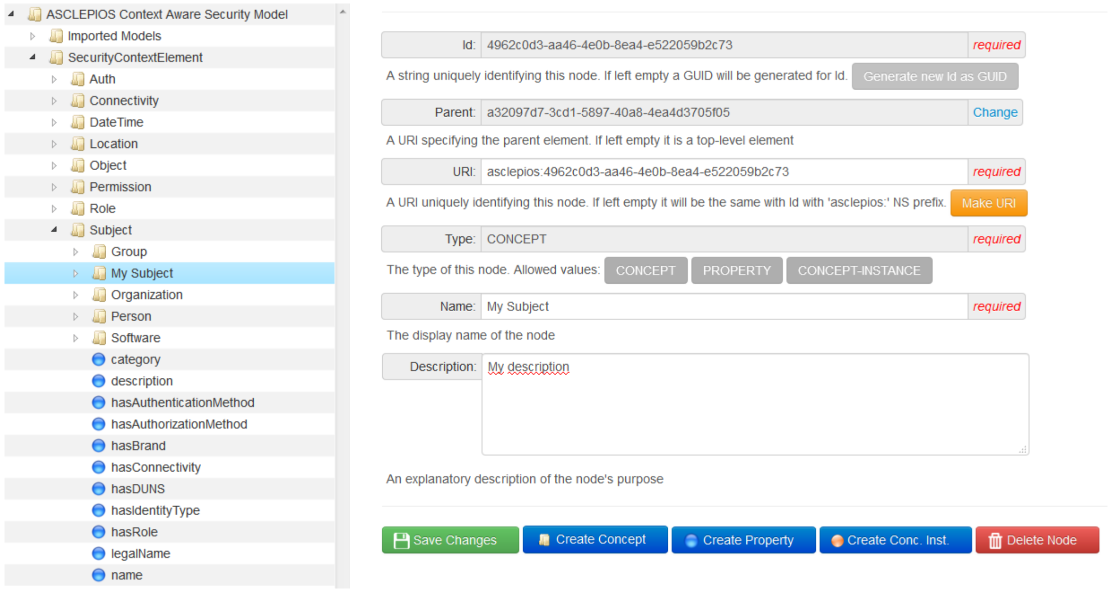
Description: An instance of this class represents the agent seeking access to a particular data artefact. This can be an organization, a person, a group or a service. Extends foaf:Agent http://purl.org/goodrelations/v1#BusinessEntity http://purl.org/goodrelations/v1#ProductOrService

An explanatory description of the node's purpose

Figure 16 – New attribute in CASM

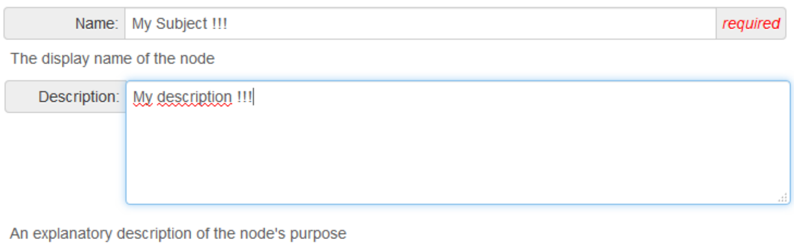
Edit an existing attribute

Selecting an element in the CASM tree on the left, will load its details in the details form on the right (see Figure 17). The user has the opportunity to modify the information in the form (except *Id*). In case of attributes the name and description fields are the most commonly updated fields (see Figure 18).



Id: 4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.
Parent: a32097d7-3cd1-5897-40a8-4ea4d3705f05
A URI specifying the parent element. If left empty it is a top-level element
URI: asclepios:4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.
Type: CONCEPT *required*
The type of this node. Allowed values:
Name: My Subject *required*
The display name of the node
Description: My description
An explanatory description of the node's purpose

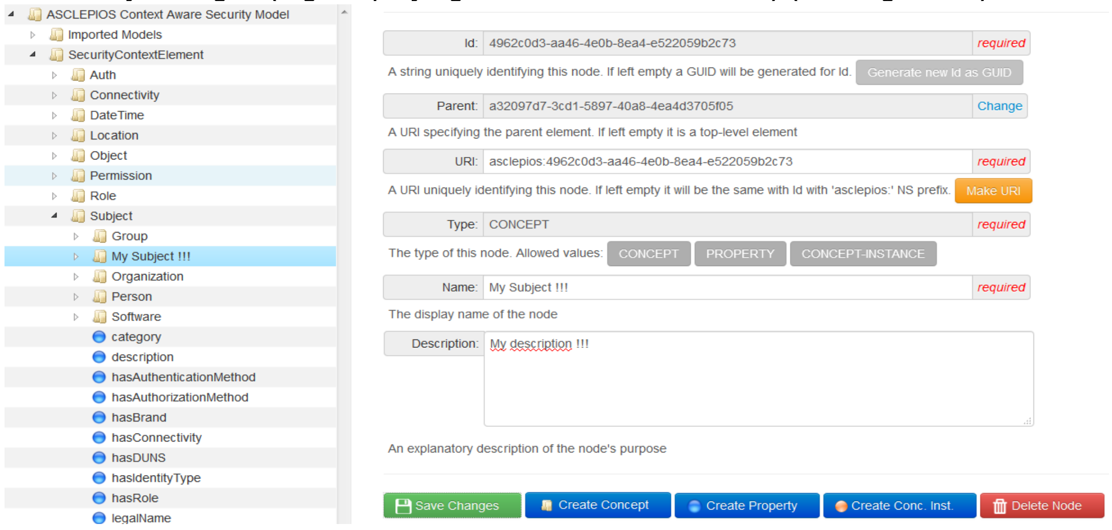
Figure 17 – Edit an existing attribute



Name: My Subject !!! *required*
The display name of the node
Description: My description !!!
An explanatory description of the node's purpose

Figure 18 – Change an attribute's details

After saving changes, the CASM tree on the left-hand side of the page will refresh in order to reflect any changes (e.g. displaying the new attribute name) (see Figure 19).

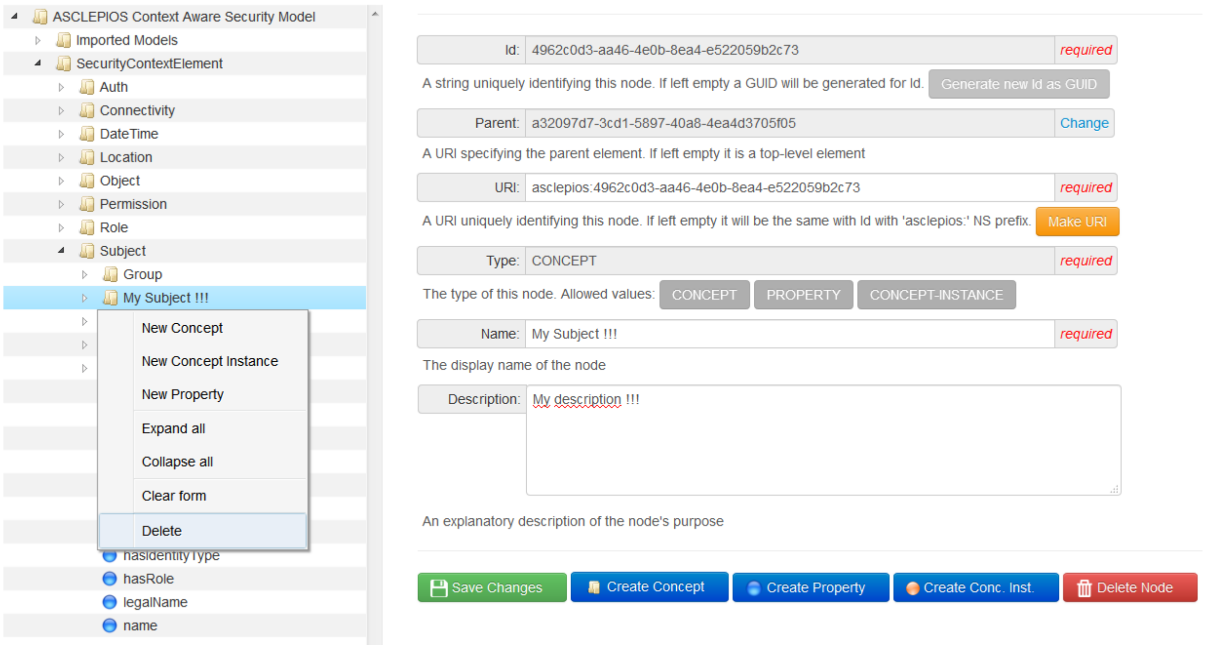


Id: 4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.
Parent: a32097d7-3cd1-5897-40a8-4ea4d3705f05
A URI specifying the parent element. If left empty it is a top-level element
URI: asclepios:4962c0d3-aa46-4e0b-8ea4-e522059b2c73 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.
Type: CONCEPT *required*
The type of this node. Allowed values:
Name: My Subject !!! *required*
The display name of the node
Description: My description !!!
An explanatory description of the node's purpose

Figure 19 – Changed attribute in CASM

Delete an attribute

Selecting an element in the CASM tree on the left, will load its details in the details form on the right. The user has the option to delete that particular element and its child elements. This can be achieved by pressing the *Delete* button residing under details form or using the corresponding option of the context menu, accessible with right-click (see Figure 20). In both cases the user will need to confirm operation before deletion is carried out (see Figure 21).



Id: 4962c0d3-aa46-4e0b-8ea4-e522059b2c73 required
A string uniquely identifying this node. If left empty a GUID will be generated for Id. Generate new Id as GUID
Parent: a32097d7-3cd1-5897-40a8-4ea4d3705f05 Change
A URI specifying the parent element. If left empty it is a top-level element
URI: asclepios:4962c0d3-aa46-4e0b-8ea4-e522059b2c73 required
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix. Make URI
Type: CONCEPT required
The type of this node. Allowed values: CONCEPT PROPERTY CONCEPT-INSTANCE
Name: My Subject !!! required
The display name of the node
Description: My description !!!
An explanatory description of the node's purpose

Figure 20 – Deleting an attribute using context menu

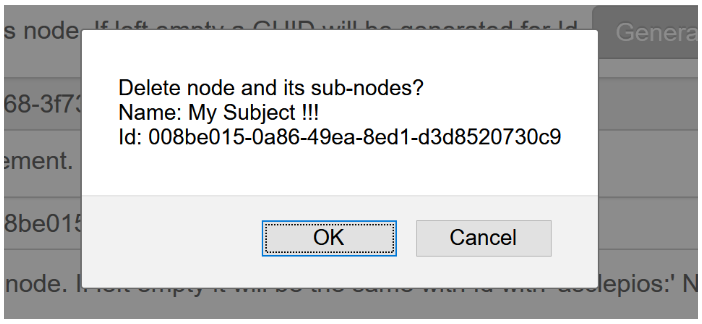

Delete node and its sub-nodes?
Name: My Subject !!!
Id: 008be015-0a86-49ea-8ed1-d3d8520730c9

Figure 21 – Confirm attribute deletion

I.3. CASM Editor REST API

Apart from a graphical user interface, CASM Editor encompasses a REST API controller for providing its functionality through REST calls. The relevant endpoints, operations and message formats are detailed in the following table.

Table 4 – REST API of CASM Editor

REST endpoint and Verb	Request	Response

getTopLevelAttributes
`GET /opt/attributes/`
 	
Description: Returns an array with all top-level attributes of CASM. The full descriptions of attributes are contained in the array, as shown in the example at the Response column.
	n/a	application/json
		<SchemaObject> array
		
		`[`
		`  {`
		`    "id": "string",`
		`    "name": "string",`
		`    "type": "string",`
		`    "uri": "string",`
		`    "description": "string",`
		`    "createTimestamp":`
		`        "2020-03-05T07:13:43.573Z",`
		`    "lastUpdateTimestamp":`
		`        "2020-03-05T07:13:43.573Z",`
		`    "propertyIsA": "string",`
		`    "propertyIsA_display": "string",`
		`    "propertyType": "string",`
		`    "propertyValue": "string",`
		`    "range": "string",`
		`    "rangeUri": "string",`
		`    "range_display": "string"`
		`  }`
		`]`

getAllAttributes
`GET /opt/attributes/all`
 	
Description: Returns an array containing the full descriptions of all attributes, properties and instances in CASM.
	`n/a`	`application/json`
		`<SchemaObject> array`
		
		For an example see at **getTopLevelAttributes**

getAttribute
`GET /opt/attributes/{attr_id}`
`attr_id : String`
 	
Description: Returns the description of the attribute or element, specified by `attr_id` parameter, which matches to the `id` property of the attribute.
	`n/a`	`application/json`
		`SchemaObject`
		
		`{`
		`  "id": "string",`
		`  "name": "string",`
		`  "type": "string",`
		`  "uri": "string",`
		`  "description": "string",`
		`  "createTimestamp":`
		`        "2020-03-05T07:17:27.295Z",`
		`  "lastUpdateTimestamp":`
		`        "2020-03-05T07:17:27.295Z",`
		`  "propertyIsA": "string",`
		`  "propertyIsA_display": "string",`
		`  "propertyType": "string",`
		`  "propertyValue": "string",`
		`  "range": "string",`
		`  "rangeUri": "string",`
		`  "range_display": "string"`
		`}`

getAttributeSubattributes
`GET /opt/attributes/{attr_id}/subattributes`
`attr_id : String`
 	
Description: Returns an array with all child elements of the attribute specified by `attr_id` parameter, which matches to the `id` property of the attribute.
	`n/a`	`application/json`
		`<SchemaObject> array`
		
		See at **getTopLevelAttributes**

findAttributesByName
`GET /opt/attributes/search/by-name/{term}`
`term : String`
 	
Description: Returns an array with all attributes whose names contain the term specified by `term` parameter. Search is case insensitive.
	`n/a`	`application/json`
		`<SchemaObject> array`
		
		See at **getTopLevelAttributes**

findPropertiesByAttribute
`GET /opt/attributes/search/properties/by-attribute/{attr_id}`
`attr_id : String`
 	
Description: Returns an array containing all properties belonging (as child elements) to the attribute specified by `attr_id` parameter, which matches to the `id` property of the attribute.
	`n/a`	`application/json`
		`<SchemaObject> array`
		
		See at **getTopLevelAttributes**

createAttribute `PUT /opt/attributes/` **Description:** Creates a new attribute or property or instance in CASM. It requires the full definition of the new element in the Request body. It returns a plain text message with the result of the operation.	`application/json` `SchemaObject` `{ "id": "string", "name": "string", "type": "string", "uri": "string", "description": "string", "createTimestamp": "2020-03-05T07:17:27.295Z", "lastUpdateTimestamp": "2020-03-05T07:17:27.295Z", "propertyIsA": "string", "propertyIsA_display": "string", "propertyType": "string", "propertyValue": "string", "range": "string", "rangeUri": "string", "range_display": "string" }`	`text/plain` `string`
updateAttribute `POST /opt/attributes/{attr_id}` `attr_id : String` **Description:** Replaces the definition of an existing element, specified by `attr_id` parameter, with a new definition. It requires the full (new) definition in the Request body. It returns a plain text message with the result of the operation.	`application/json` `SchemaObject` See at **createAttribute**	`text/plain` `string`

deleteAttribute
`DELETE /opt/attributes/{attr_id}`
`attr_id : String`
 	
Description: Deletes an existing element, specified by `attr_id` parameter. If it is an attribute it must not have any child elements, otherwise the operation fails.
	`n/a`	text/plain
		`string`

deleteAttributeAndSubattributes
`DELETE /opt/attributes/{attr_id}/all`
`attr_id : String`
 	
Description: Deletes an existing element, specified by `attr_id` parameter, and all its child elements from CASM.
	`n/a`	text/plain
		`string`

I.4. CASM Serialization

CASM Editor stores CASM changes in Models Store as RDF triples. The following listing provides a sample excerpt from a CASM export in RDF/TTL format.

Table 5 – Sample attribute export

```
# Definition of class: EHR
<http://www.asclepios.eu/casm/ASCLEPIOS-OBJECT#1371e7a1-def9-4d84-a65e-↙
                                                                2d3060f4d97d>
    a      <http://www.asclepios.eu/casm#ASCLEPIOS-OBJECT> ;
    <http://purl.org/dc/elements/1.1/type>  "CONCEPT" ;
    <http://purl.org/dc/terms/URI>
                 "ascm:1371e7a1-def9-4d84-a65e-2d3060f4d97d" ;
    <http://purl.org/dc/terms/created>
                 "2019-12-18T19:40:55.933Z"^^<http://www.w3.org/2001/ ↙
                                            XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/description>
                 "This class represents the patient’s Electronic Health
Records (EHR). EHR represents a digital collection of medical information
about a person. It includes information about a patient???s health history,
such as diagnoses, medicines received, tests, allergies, immunizations, and
treatment plans." ;
    <http://purl.org/dc/terms/identifier>
                 "1371e7a1-def9-4d84-a65e-2d3060f4d97d" ;
    <http://purl.org/dc/terms/modified>
                 "2019-12-18T19:40:55.933Z"^^<http://www.w3.org/2001/ ↙
                                            XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/title> "EHR" ;
    <http://www.asclepios.eu/casm/types#class>
                 "eu.asclepios.ample.model.SchemaObject" ;
    <http://www.w3.org/2004/02/skos/core#broader>
                 <http://www.asclepios.eu/casm/ASCLEPIOS-OBJECT#       ↙
                                  aa205d7c-dba9-45d2-955a-d0ed0167de74> .
```

4 Attribute-Based Access Control (ABAC) Policies Editor

In this chapter the *Attribute-Based Access Control (ABAC) Policies Editor* is presented and detailed.

ABAC Policies are *sets of rules* used for making access control decisions, i.e. permitting or denying access to certain protected resources. They are evaluated (or applied) every time an access is attempted to a protected resource to yield a decision. When that happens, one, some or all policy rules are evaluated. Each policy rule, when applicable, makes a partial access control decision. The rule outcomes are then combined to yield the policy's decision, therefore an ABAC Policy must always specify a combining method (algorithm) for aggregating rules' partial decisions into the policy decision. It is worth noting that several ABAC Policies can be in effect at the same time for the same protected resources.
An ABAC Policy Rules definition must specify the rule outcome (i.e. partial access control decision) yielded, when the rule is evaluated. Rules typically include a controlling condition. When this condition holds, the rule is applicable at that particular policy application/evaluation otherwise the rule is ignored. The rule condition is a boolean expression of attribute values (like those defined in CASM) and constant values. When it evaluates to true the corresponding rule outcome is returned as a partial decision, which can subsequently be used (possibly in conjunction with other partial decisions) to render the policy decision.
In technical terms, in the context of ASCLEPIOS project, ABAC Policies and Rules are uniquely identified with an *Id* property, titled with a *Name*, and they can optionally have a *Description*. They can also be referenced using a Universal Resource Identifier (*URI*). ABAC Policies in particular must have a combining algorithm and include one or more rules. ABAC Policy rules must specify an outcome (*Permit* or *Deny*) and typically a condition controlling when they are applicable. If no rule condition is specified, the rule is always applicable. Rule conditions include CASM elements (attributes, properties, attribute instances) in their expressions.

In the following sections the ABAC Policies editor is presented in detail.

I.1. Usage Scenarios

The main goal of ABAC Policies Editor is to create and manage the ABAC Policies of ASCLEPIOS platform. This goal can be broken down to a series of specific capabilities that must be offered to the user for creating, updating, deleting and retrieving the ABAC policies along with their rules, as well as for importing and exporting them to files. A full list of ABAC Policies Editor capabilities (and related usage scenarios) is given next.

- List existing ABAC Policies and their rules
- Retrieve an ABAC Policy's details
- Create a new ABAC Policy
- Modify or Delete an existing ABAC Policy
- Retrieve an ABAC Policy Rule's details
- Create a new ABAC Policy Rule
- Modify or Delete an existing ABAC Policy Rule
- Import an ABAC Policy from a file
- Export an ABAC Policy to a file
- Interpret an ABAC Policy to XACML
- Submit an ABAC Policy to Policies Enforcement Mechanism

The ABAC Policies retrieval and editing use cases are depicted in Figure 22, ABAC Policy Rules retrieval and editing use cases are depicted in Figure 23 whereas the import/export, interpretation and submission to Policies Enforcement Mechanism use cases are depicted in Figure 24.

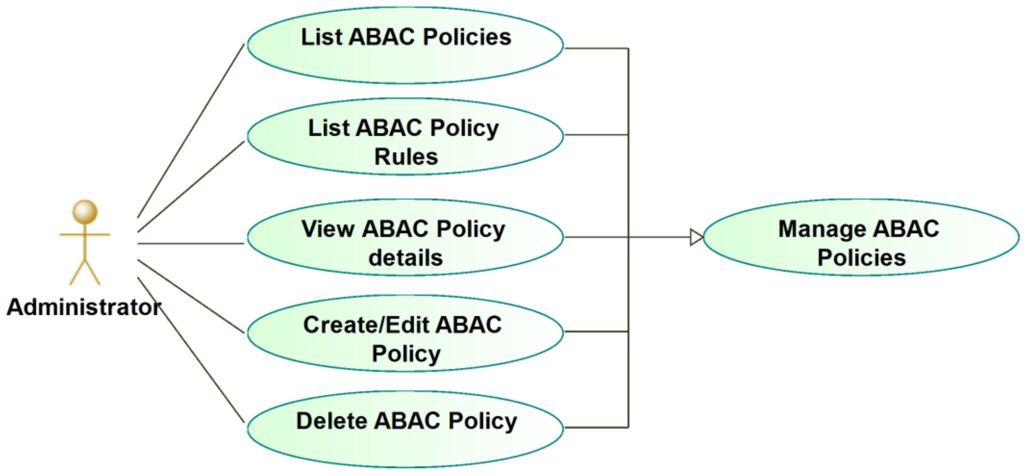


Figure 22 – ABAC Policies management use case

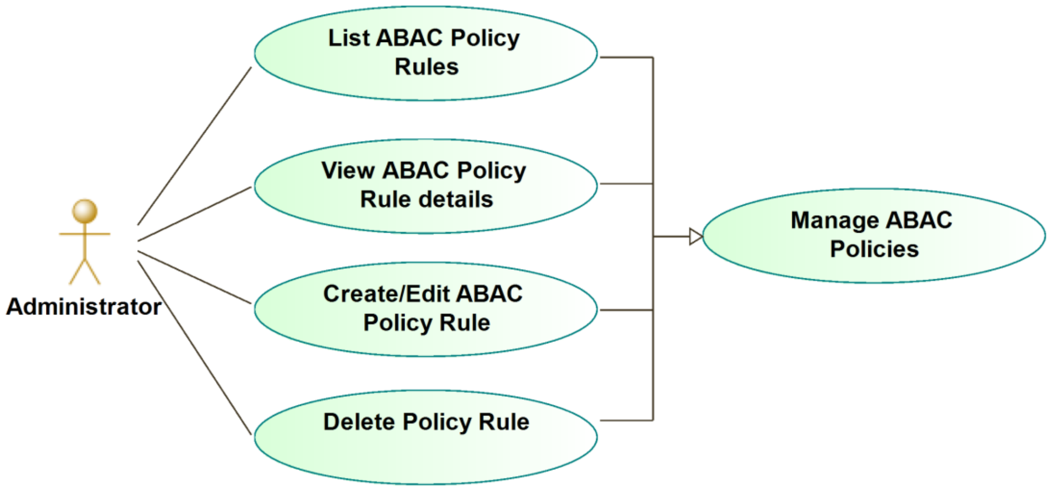


Figure 23 – ABAC Policy Rules management use case

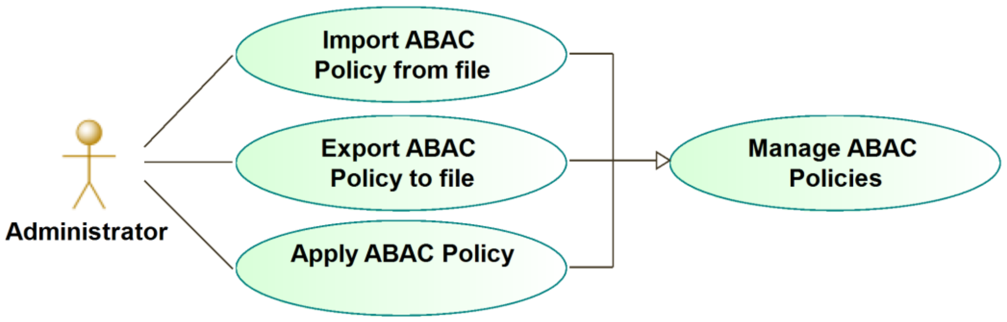


Figure 24 – ABAC Policies Import/Export, and submission to Policies Enforcement use case

I.2. Walkthrough

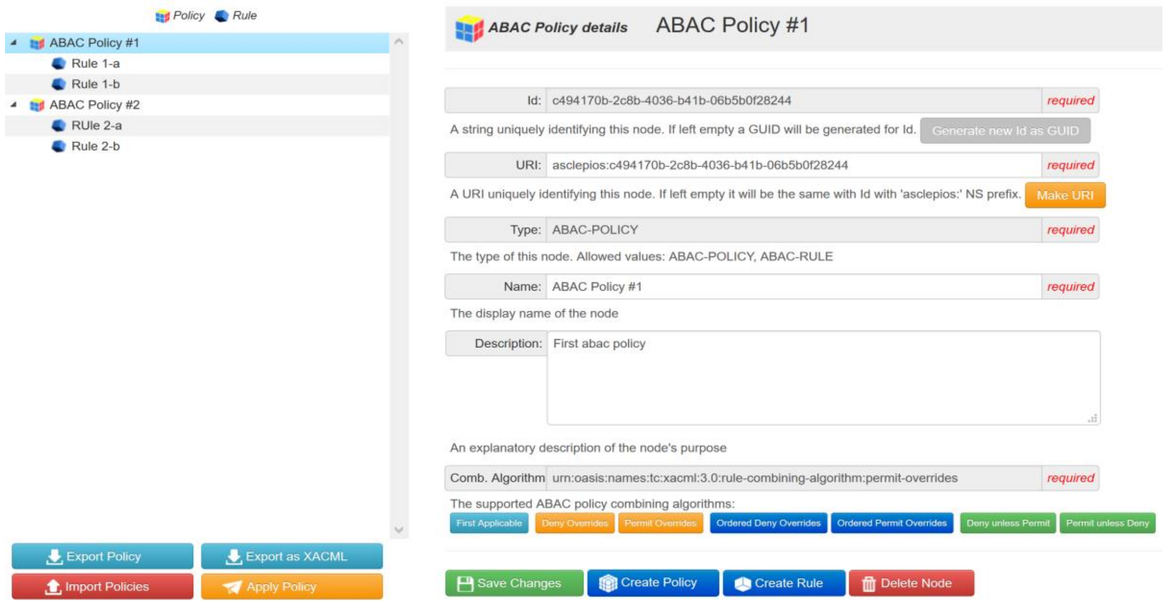
In this section the graphical user interface of the ABAC Policies Editor is briefly presented along with a short walkthrough of its operation. The editor can be accessed through AMPLE web page.

ABAC Policies Editor

ABAC Policies Editor opens by pressing the button named *"ABAC Policies"* in the Welcome screen or the respective button in menu. Figure 25 gives an overview of the ABAC Policies Editor.

The page is vertically divided in two notable regions. The left-hand part of the page contains a list of all existing ABAC Policies. The user can click on the arrow heads on the left side of each ABAC Policy in order to expand it and view its Policy Rules (if any). Clicking on an ABAC Policy or Policy Rule loads its details into the details form in the right-hand side of the page. Right clicking on an item in the list will open the context menu which offers actions related to the selected item (Figure 26).

The details form (in the right-hand side of the page) encompasses fields that are common both to ABAC Policies and Rules. Fields specific to a particular type are displayed only when an item of the corresponding type is selected. Under the details form a row of buttons exists. These buttons can be used to save any changes made in details form, create new ABAC Policies or Policy Rules or delete the currently displayed item in details form. Buttons get dimmed when the corresponding operation is not available in a particular case (e.g. when editing a Rule, creating a child Rule is not active).



Id: c494170b-2c8b-4036-b41b-06b5b0f28244

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:c494170b-2c8b-4036-b41b-06b5b0f28244

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: ABAC-POLICY

The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name: ABAC Policy #1

The display name of the node

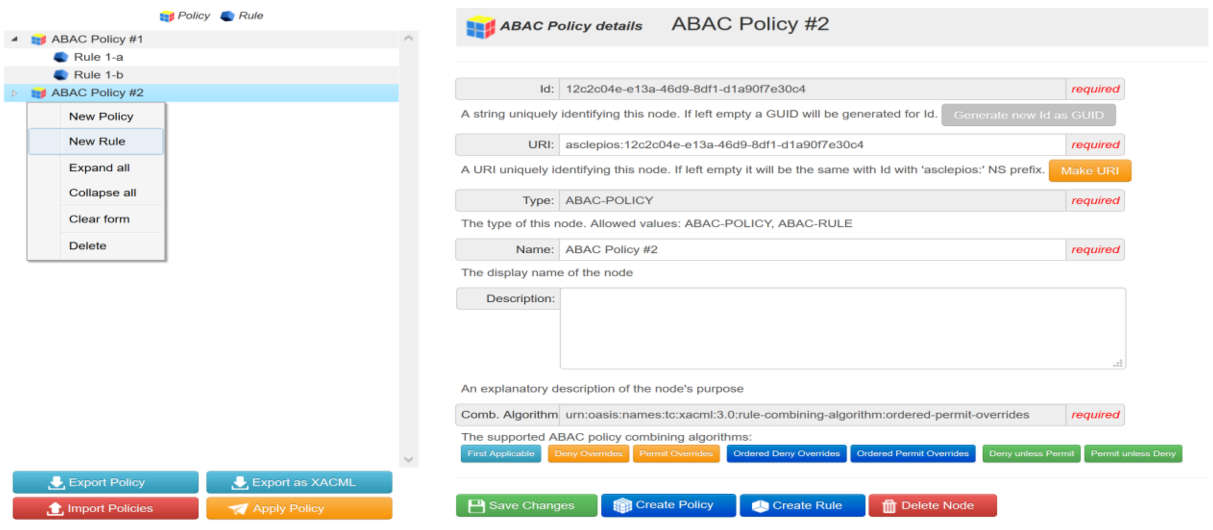
Description: First abac policy

An explanatory description of the node's purpose

Comb. Algorithm urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:permit-overrides

The supported ABAC policy combining algorithms:

Figure 25 – ABAC Policies Editor page



Id: 12c2c04e-e13a-46d9-8df1-d1a90f7e30c4

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:12c2c04e-e13a-46d9-8df1-d1a90f7e30c4

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: ABAC-POLICY

The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name: ABAC Policy #2

The display name of the node

Description:

An explanatory description of the node's purpose

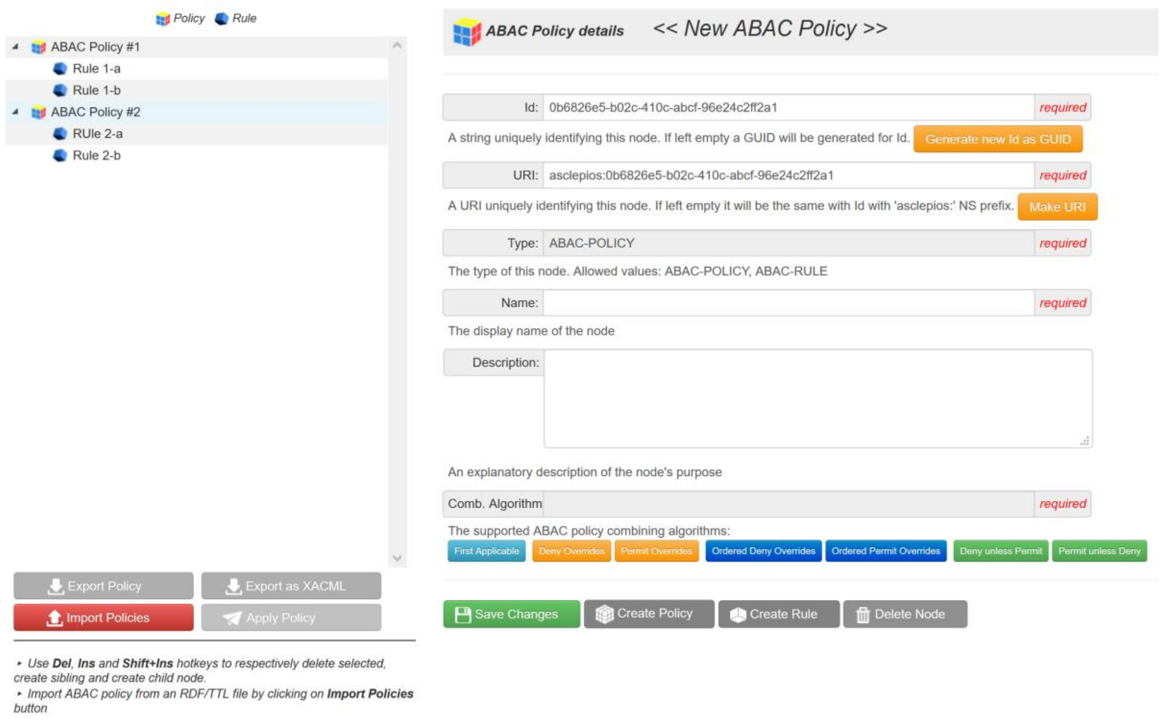
Comb. Algorithm urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:ordered-permit-overrides

The supported ABAC policy combining algorithms:

Figure 26 – Creation of a new rule using context menu

Create new ABAC Policy

The user can create a new ABAC Policy by clicking on the *"Create Policy"* button under the details form (see Figure 27). This action will clear details form, and prefill *Id*, *Type* and *URI* fields with new values. The user has the opportunity to modify their values before saving them (Figure 28). When ready user can click on *Save* button (colored green) to submit information to the server for saving.



<< New ABAC Policy >>

Id:	0b6826e5-b02c-410c-abcf-96e24c2ff2a1	*required*

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI:	asclepios:0b6826e5-b02c-410c-abcf-96e24c2ff2a1	*required*

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type:	ABAC-POLICY	*required*

The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name:		*required*

The display name of the node

Description:

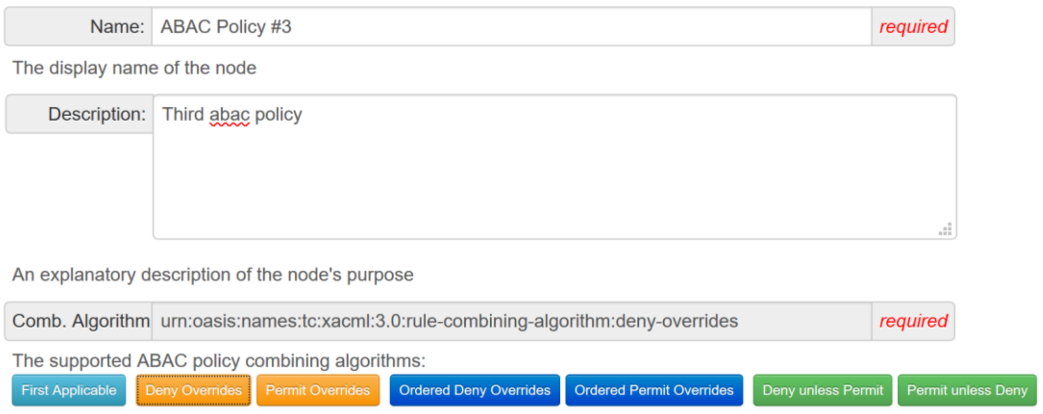
An explanatory description of the node's purpose

Comb. Algorithm		*required*

The supported ABAC policy combining algorithms:

- *Use **Del**, **Ins** and **Shift+Ins** hotkeys to respectively delete selected, create sibling and create child node.*
- *Import ABAC policy from an RDF/TTL file by clicking on **Import Policies** button*

Figure 27 – New ABAC Policy form



Name:	ABAC Policy #3	*required*

The display name of the node

Description:	Third abac policy

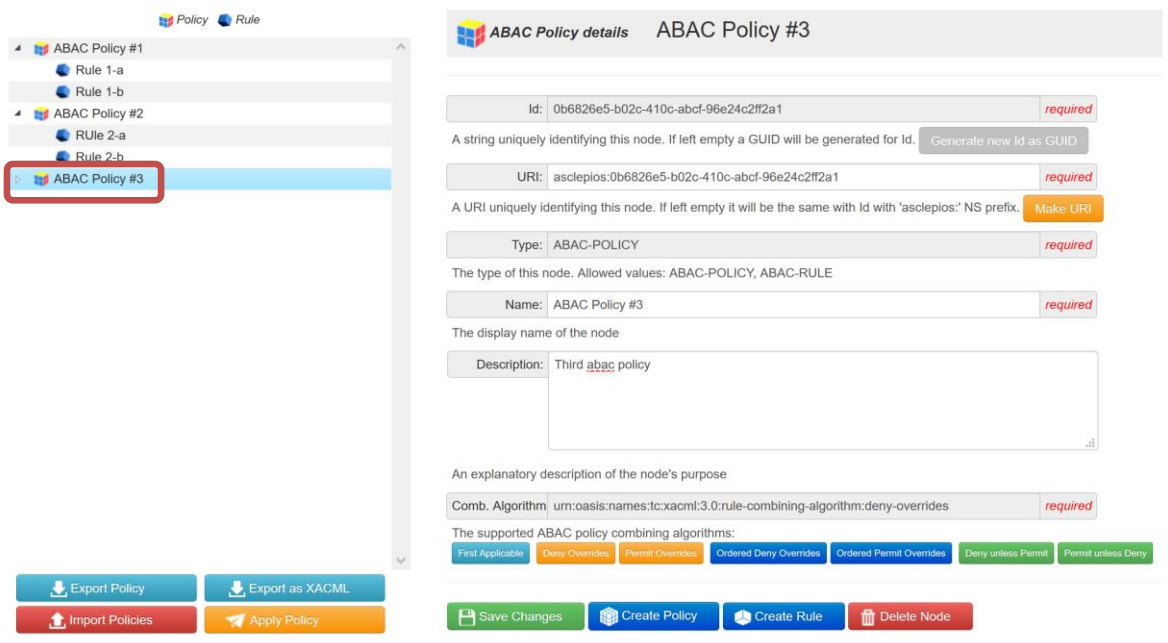
An explanatory description of the node's purpose

Comb. Algorithm	urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides	*required*

The supported ABAC policy combining algorithms:

Figure 28 – Filling in new ABAC Policy details

After saving changes, the ABAC Policies list on the left-hand side of the page will refresh in order to reflect the changes, i.e. including the new ABAC Policy (Figure 29).



ABAC Policy details ABAC Policy #3

Id:	0b6826e5-b02c-410c-abcf-96e24c2ff2a1	required
URI:	asclepios:0b6826e5-b02c-410c-abcf-96e24c2ff2a1	required
Type:	ABAC-POLICY	required
Name:	ABAC Policy #3	required
Description:	Third abac policy	
Comb. Algorithm	urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides	required

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

The display name of the node

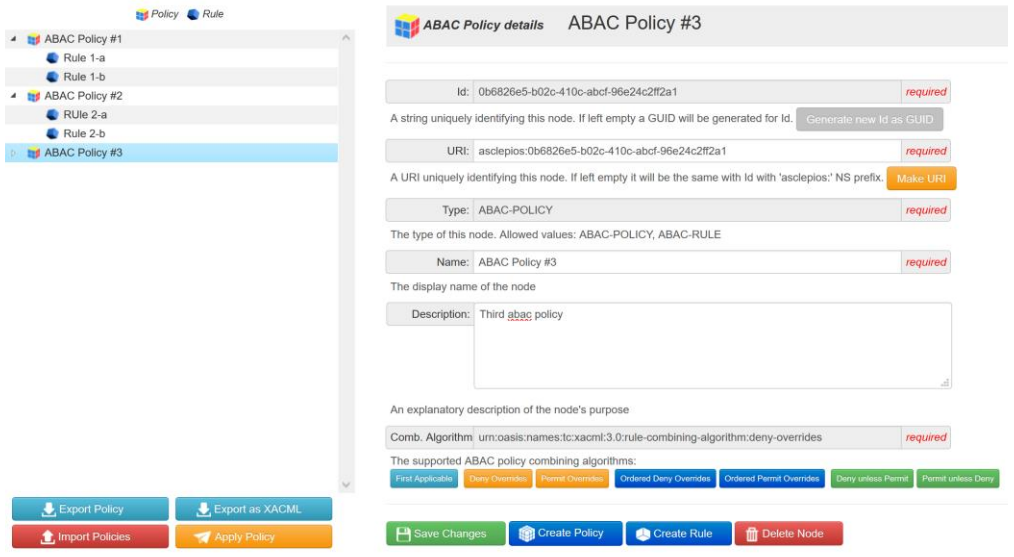
An explanatory description of the node's purpose

The supported ABAC policy combining algorithms:

Figure 29 – New ABAC Policy

Edit an existing ABAC Policy

Selecting an ABAC Policy in the list on the left will load its information in the details form on the right (see Figure 30). User has the option to modify the information in the form (except *Id* and *Type*) (see Figure 31).



ABAC Policy details ABAC Policy #3

Id:	0b6826e5-b02c-410c-abcf-96e24c2ff2a1	required
URI:	asclepios:0b6826e5-b02c-410c-abcf-96e24c2ff2a1	required
Type:	ABAC-POLICY	required
Name:	ABAC Policy #3	required
Description:	Third abac policy	
Comb. Algorithm	urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-overrides	required

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

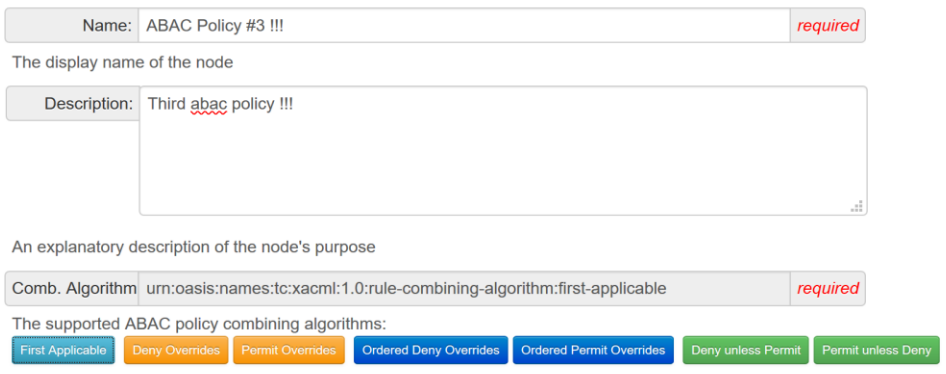
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

The display name of the node

An explanatory description of the node's purpose

The supported ABAC policy combining algorithms:

Figure 30 – Edit an existing ABAC Policy



Name:	ABAC Policy #3 !!!	required
Description:	Third abac policy !!!	
Comb. Algorithm	urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable	required

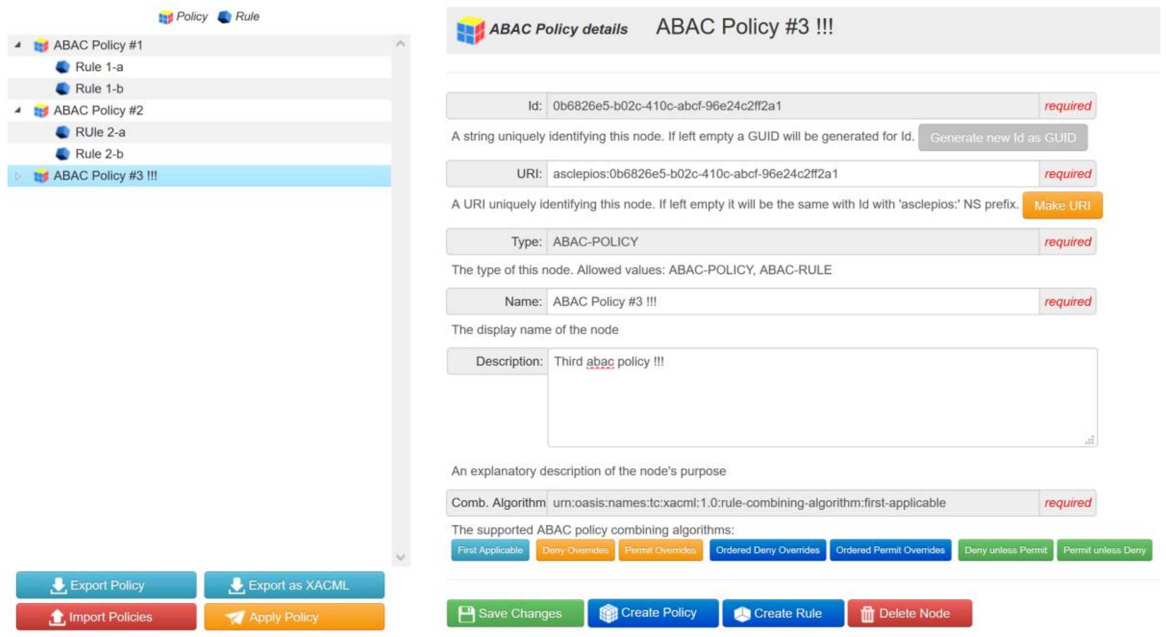
The display name of the node

An explanatory description of the node's purpose

The supported ABAC policy combining algorithms:

Figure 31 – Change an ABAC Policy's details

After saving changes, the list on the left-hand side of the page will refresh in order to reflect any changes (e.g. displaying the new ABAC Policy name) (see Figure 32).



ABAC Policy details ABAC Policy #3 !!!

Id: 0b6826e5-b02c-410c-abcf-96e24c2ff2a1 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:0b6826e5-b02c-410c-abcf-96e24c2ff2a1 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: ABAC-POLICY *required*
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name: ABAC Policy #3 !!! *required*
The display name of the node

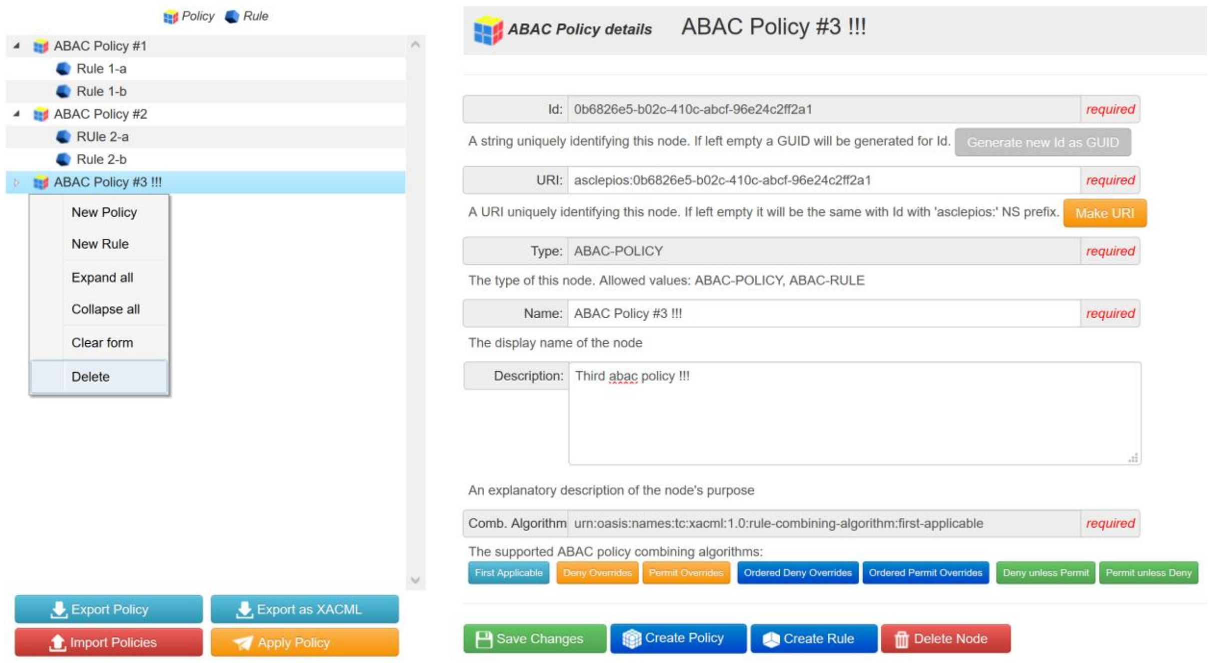
Description: Third abac policy !!!
An explanatory description of the node's purpose

Comb. Algorithm: urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable *required*
The supported ABAC policy combining algorithms:

Figure 32 – Changed ABAC Policy in the list

Delete an ABAC Policy

Selecting an ABAC Policy in the list on the left will load its details in the details form on the right. User has the option to delete that particular ABAC Policy and its rules. This can be achieved by pressing the *Delete* button residing under details form or using the corresponding option of the context menu, accessible with right-click (see Figure 33). In both cases the user will need to confirm operation before deletion is carried out (see Figure 34).



ABAC Policy details ABAC Policy #3 !!!

Id: 0b6826e5-b02c-410c-abcf-96e24c2ff2a1 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:0b6826e5-b02c-410c-abcf-96e24c2ff2a1 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

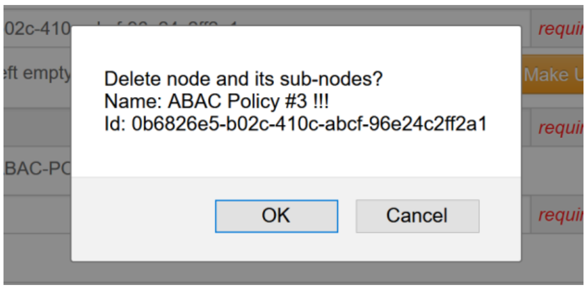
Type: ABAC-POLICY *required*
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name: ABAC Policy #3 !!! *required*
The display name of the node

Description: Third abac policy !!!
An explanatory description of the node's purpose

Comb. Algorithm: urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable *required*
The supported ABAC policy combining algorithms:

Figure 33 – Deleting an ABAC Policy using context menu

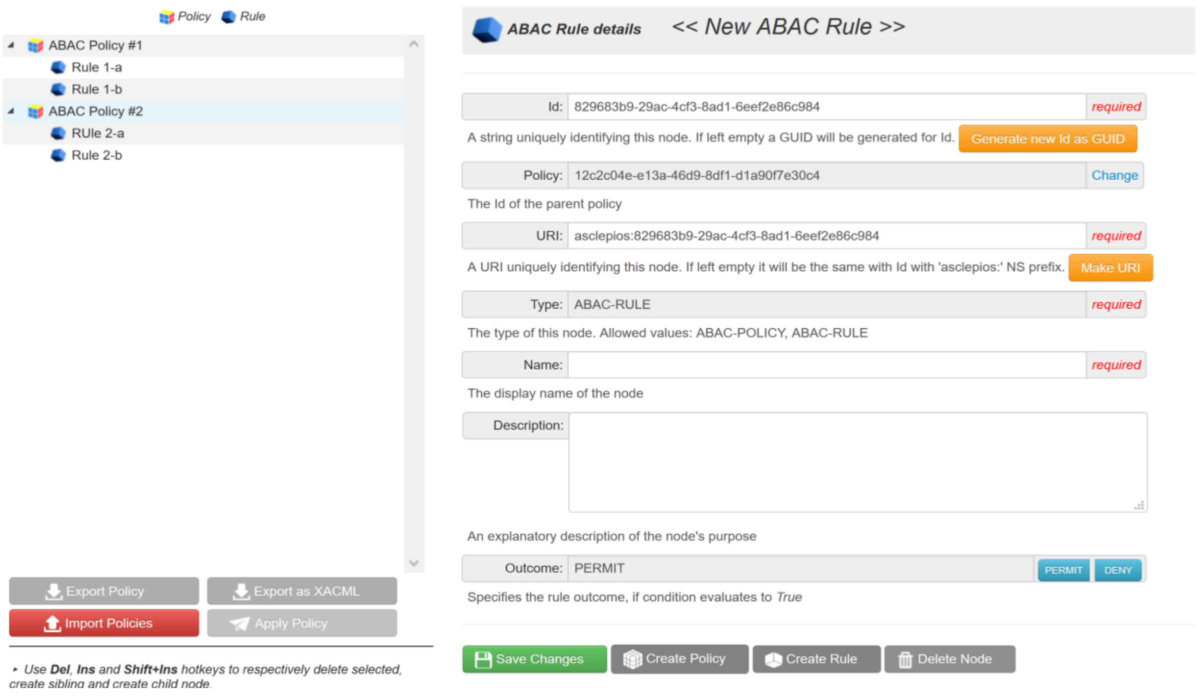


Delete node and its sub-nodes?
Name: ABAC Policy #3 !!!
Id: 0b6826e5-b02c-410c-abcf-96e24c2ff2a1

Figure 34 – Confirm ABAC Policy deletion

Create new ABAC Policy Rule

User can create a new ABAC Policy Rule by clicking on the *"Create Rule"* button under the details form. This action will clear details form, and prefill *Id*, *Policy* and *URI* fields with new values (see Figure 35). User has the opportunity to modify their values before saving them (Figure 36). When ready user can click on *Save* button (colored green) to submit information to the server for saving.



▸ *Use **Del**, **Ins** and **Shift+Ins** hotkeys to respectively delete selected, create sibling and create child node.*

ABAC Rule details *<< New ABAC Rule >>*

Id:	829683b9-29ac-4cf3-8ad1-6eef2e86c984	*required*
Policy:	12c2c04e-e13a-46d9-8df1-d1a90f7e30c4	Change
URI:	asclepios:829683b9-29ac-4cf3-8ad1-6eef2e86c984	*required*
Type:	ABAC-RULE	*required*
Name:		*required*
Description:		
Outcome:	PERMIT	PERMIT DENY

A string uniquely identifying this node. If left empty a GUID will be generated for Id. Generate new Id as GUID

The Id of the parent policy

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix. Make URI

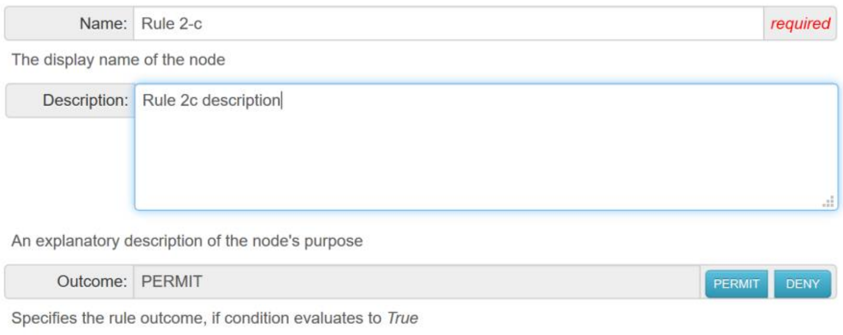
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

The display name of the node

An explanatory description of the node's purpose

Specifies the rule outcome, if condition evaluates to *True*

Figure 35 – New ABAC Policy Rule form



Name:	Rule 2-c	*required*
Description:	Rule 2c description	
Outcome:	PERMIT	PERMIT DENY

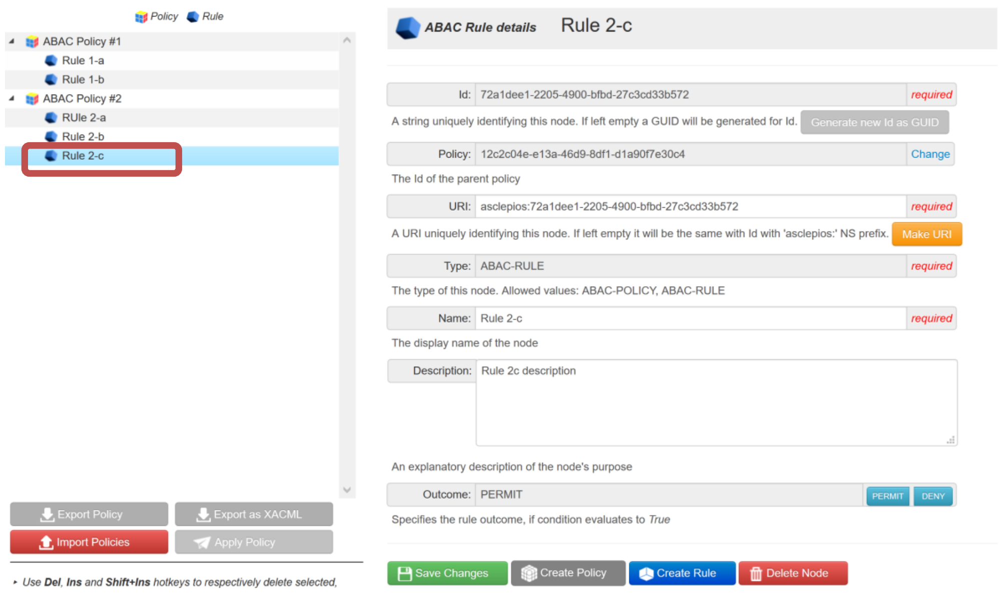
The display name of the node

An explanatory description of the node's purpose

Specifies the rule outcome, if condition evaluates to *True*

Figure 36 – Filling in new ABAC Policy Rule details

After saving changes, the ABAC Policies list on the left-hand side of the page will refresh in order to reflect the changes, i.e. including the new ABAC Policy (Figure 37).


Policy Rule

- ABAC Policy #1
  - Rule 1-a
  - Rule 1-b
- ABAC Policy #2
  - RUle 2-a
  - Rule 2-b
  - Rule 2-c

Export Policy | Export as XACML | Import Policies | Apply Policy

▸ *Use **Del**, **Ins** and **Shift+Ins** hotkeys to respectively delete selected,*

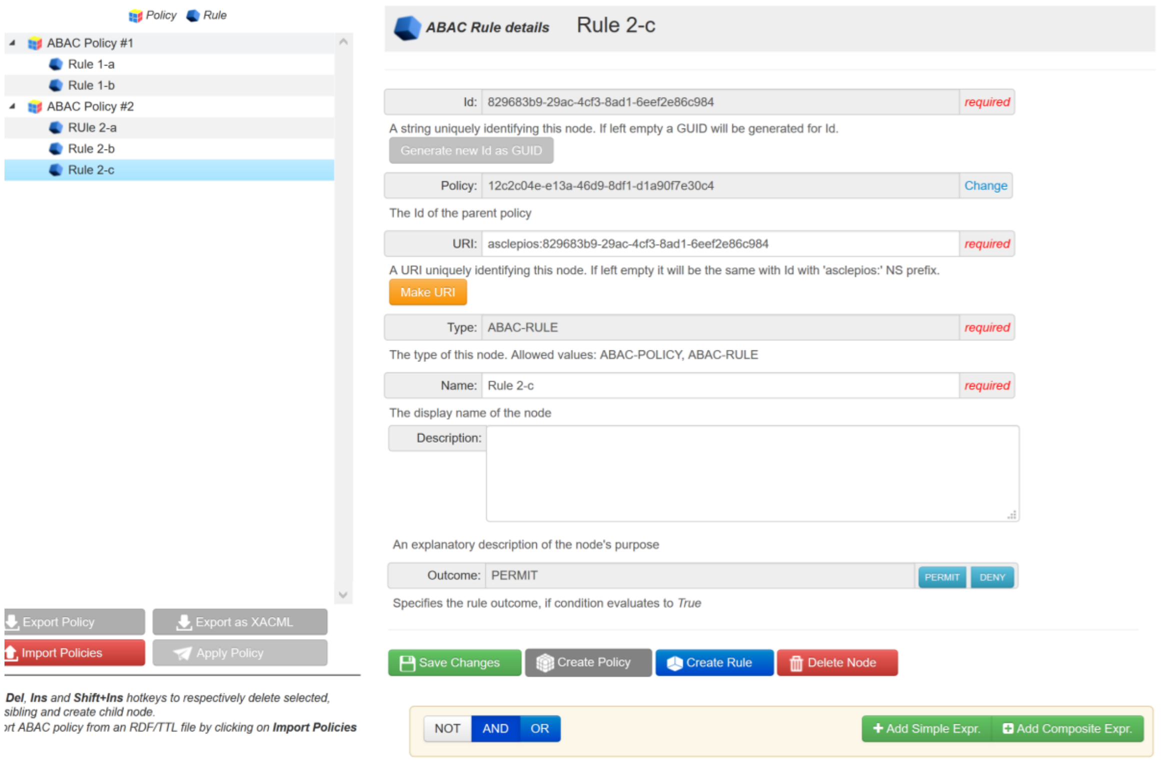
ABAC Rule details Rule 2-c

Id:	72a1dee1-2205-4900-bfbd-27c3cd33b572	*required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.		Generate new Id as GUID
Policy:	12c2c04e-e13a-46d9-8df1-d1a90f7e30c4	Change
The Id of the parent policy		
URI:	asclepios:72a1dee1-2205-4900-bfbd-27c3cd33b572	*required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.		Make URI
Type:	ABAC-RULE	*required*
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE		
Name:	Rule 2-c	*required*
The display name of the node		
Description:	Rule 2c description	
An explanatory description of the node's purpose		
Outcome:	PERMIT	PERMIT DENY
Specifies the rule outcome, if condition evaluates to *True*		

Save Changes | Create Policy | Create Rule | Delete Node

Figure 37 – New ABAC Policy Rule in the list

An integral part of an ABAC Policy Rule is its *Condition*, which is a boolean expression of attributes, instances, properties, as they have been specified in CASM. The condition controls whether a rule can be applied, when evaluating an ABAC Policy in order to grant or deny access to a resource. When condition evaluates to *true* the specified rule outcome (in details form) is returned, otherwise the rule is ignored.
The graphical expression builder, which resides at the right of the details form, can be used in order to create the rule condition (Figure 38). The expression builder provides dynamic lists (in the form of combo boxes) for the easier search and selection of CASM elements required for building the condition.
Rule condition is saved along with the details form information, when the *Save* button is pressed.


Policy Rule

- ABAC Policy #1
  - Rule 1-a
  - Rule 1-b
- ABAC Policy #2
  - RUle 2-a
  - Rule 2-b
  - Rule 2-c

Export Policy | Export as XACML | Import Policies | Apply Policy

__Del__, __Ins__ and __Shift+Ins__ hotkeys to respectively delete selected,
sibling and create child node.
ɔrt ABAC policy from an RDF/TTL file by clicking on __Import Policies__

ABAC Rule details Rule 2-c

Id:	829683b9-29ac-4cf3-8ad1-6eef2e86c984	*required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.		Generate new Id as GUID
Policy:	12c2c04e-e13a-46d9-8df1-d1a90f7e30c4	Change
The Id of the parent policy		
URI:	asclepios:829683b9-29ac-4cf3-8ad1-6eef2e86c984	*required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.		Make URI
Type:	ABAC-RULE	*required*
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE		
Name:	Rule 2-c	*required*
The display name of the node		
Description:		
An explanatory description of the node's purpose		
Outcome:	PERMIT	PERMIT DENY
Specifies the rule outcome, if condition evaluates to *True*		

Save Changes | Create Policy | Create Rule | Delete Node

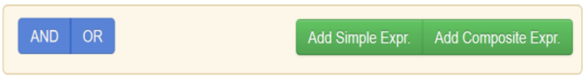
NOT | AND | OR | + Add Simple Expr. | Add Composite Expr.

Figure 38 – ABAC Policy Rule condition expression editor

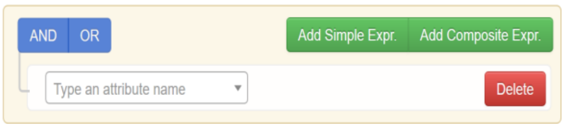
Initially the rule condition is empty. Pressing the *Add Simple Expr.* button (on the right hand side) a new expression is started containing a simple expression clause. More clauses can be added in the following lines by pressing the same button. Each line contains the definition of a simple expression clause of the form: `<Attribute>` `<Property>` `<Value>`
For example: `NetworkLocation` `hasSubnet` `"10.10.0.0"`

Clauses are combined using the boolean operator indicated by the selected button on the to top left corner of the expression editor (AND, OR). Subexpressions can be added by pressing the *Add Composite Expr.* button. Sub-expressions have their own boolean operator and simple clauses. The following screenshots give an impression of expression editor (Figure 39).

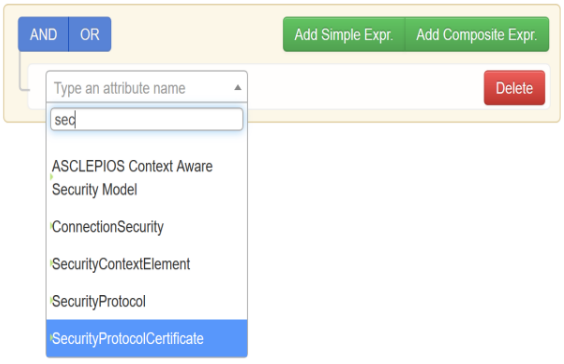
Empty rule conditions (default)



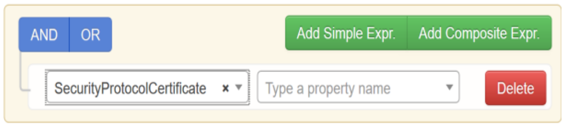
Pressing "Add Simple Expr." a new condition is added



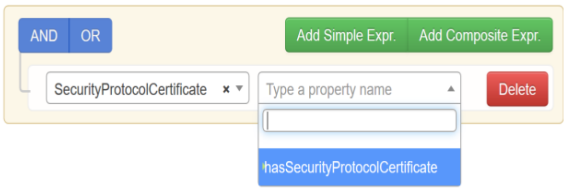
Using selection box an attribute can be selected



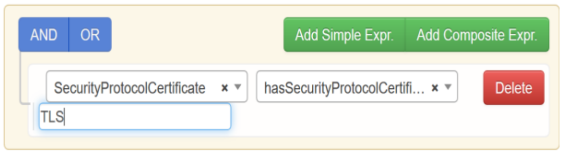
Properties selection box appears



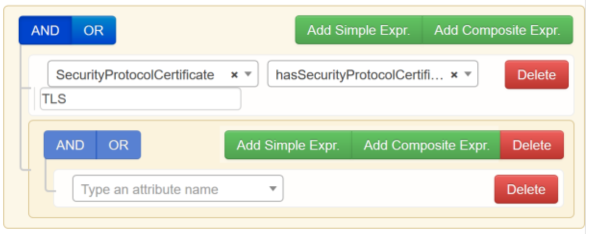
Using second selection box
an attribute's property can be selected



Property value input appears



Pressing "Add Composite Expr."
a new sub-expression is added



Example condition
with sub-expression

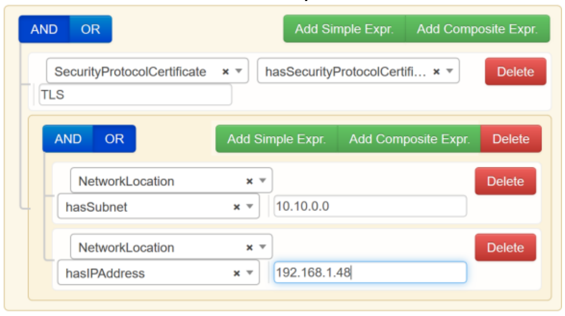
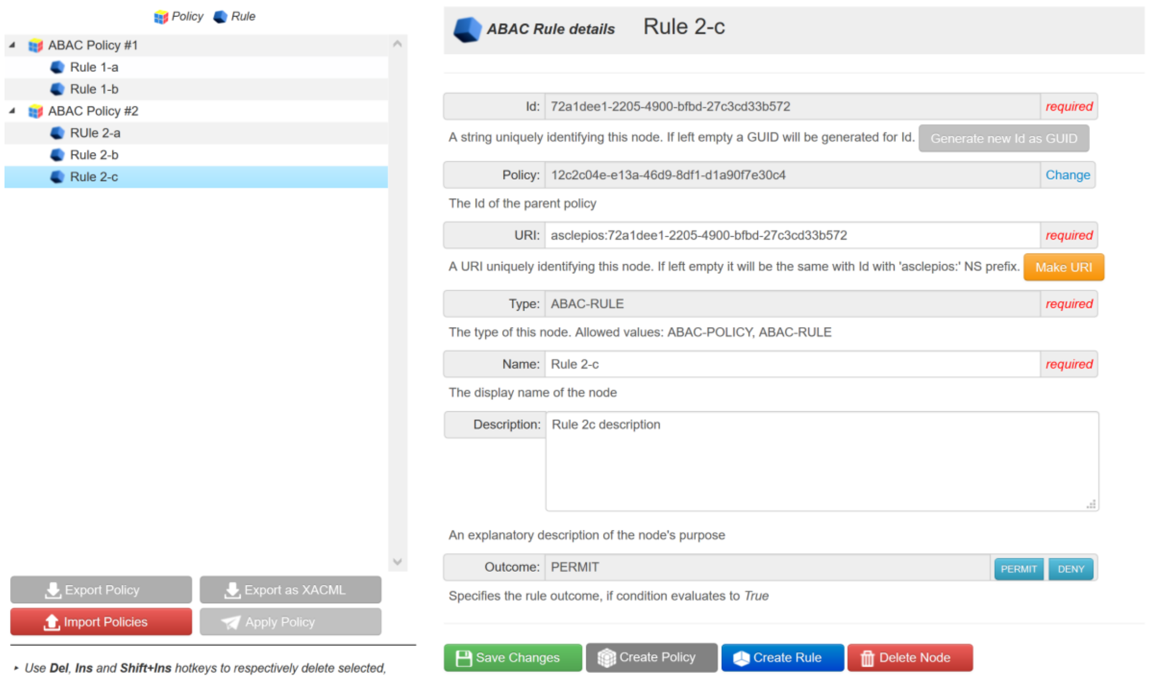


Figure 39 – Rule condition editing with expression builder

Edit an existing ABAC Policy Rule

Selecting an ABAC Policy Rule in the list on the left will load its information in the details form on the right (see Figure 40). User has the option to modify the information in the form (except *Id* and *Type*) (see Figure 41). Rule condition can also be updated.



Rule 2-c

Id:	72a1dee1-2205-4900-bfbd-27c3cd33b572	*required*
Policy:	12c2c04e-e13a-46d9-8df1-d1a90f7e30c4	Change
URI:	asclepios:72a1dee1-2205-4900-bfbd-27c3cd33b572	*required*
Type:	ABAC-RULE	*required*
Name:	Rule 2-c	*required*
Description:	Rule 2c description	
Outcome:	PERMIT	

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

The Id of the parent policy

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

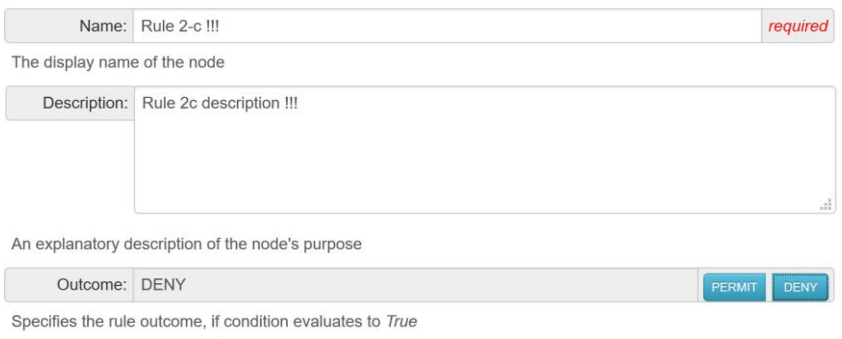
The display name of the node

An explanatory description of the node's purpose

Specifies the rule outcome, if condition evaluates to *True*

▸ *Use **Del**, **Ins** and **Shift+Ins** hotkeys to respectively delete selected,*

Figure 40 – Edit an existing ABAC Policy Rule



Name:	Rule 2-c !!!	*required*
Description:	Rule 2c description !!!	
Outcome:	DENY	

The display name of the node

An explanatory description of the node's purpose

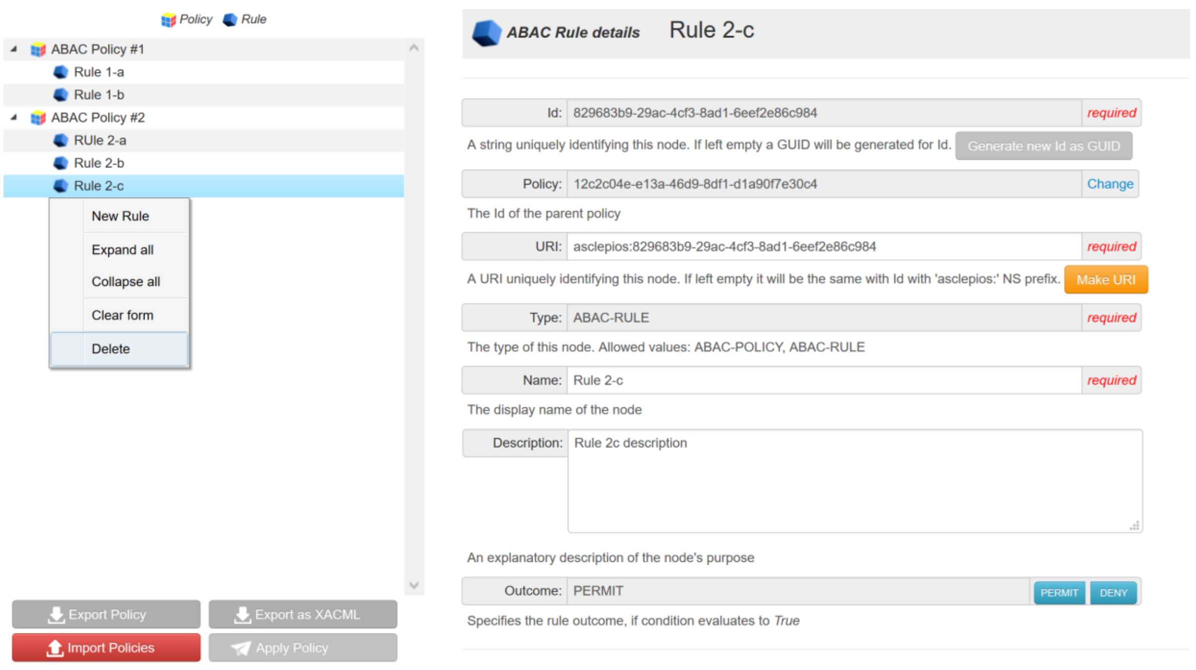
Specifies the rule outcome, if condition evaluates to *True*

Figure 41 – Change an ABAC Policy Rule's details

After saving changes, the list on the left-hand side of the page will refresh in order to reflect any changes (e.g. displaying the new ABAC Policy Rule name).

Delete an ABAC Policy Rule

Selecting an ABAC Policy Rule in the list on the left will load its details in the details form on the right. User has the opportunity to delete that particular ABAC Policy Rule, including its rule condition. This can be achieved by pressing the *Delete* button residing under details form or using the corresponding option of the context menu, accessible with right-click (see Figure 42). In both cases the user will need to confirm operation before deletion is carried out (see Figure 43).



ABAC Rule details Rule 2-c

Id: 829683b9-29ac-4cf3-8ad1-6eef2e86c984 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.

Policy: 12c2c04e-e13a-46d9-8df1-d1a90f7e30c4
The Id of the parent policy

URI: asclepios:829683b9-29ac-4cf3-8ad1-6eef2e86c984 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

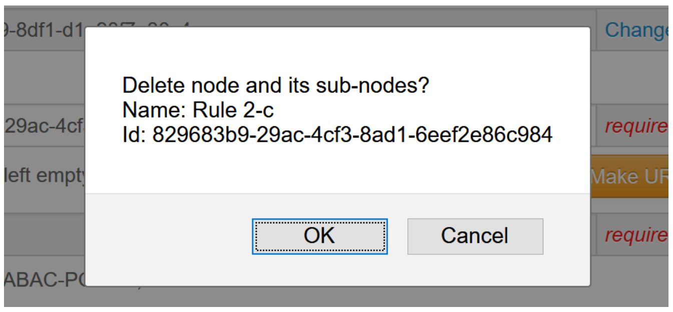
Type: ABAC-RULE *required*
The type of this node. Allowed values: ABAC-POLICY, ABAC-RULE

Name: Rule 2-c *required*
The display name of the node

Description: Rule 2c description
An explanatory description of the node's purpose

Outcome: PERMIT
Specifies the rule outcome, if condition evaluates to *True*

Figure 42 – Deleting an ABAC Policy Rule using context menu



Delete node and its sub-nodes?
Name: Rule 2-c
Id: 829683b9-29ac-4cf3-8ad1-6eef2e86c984

Figure 43 – Confirm ABAC Policy Rule deletion

Import/Export an ABAC Policy

It is possible to import and export ABAC policies as RDF/TTL files for backup or migration purposes (Figure 44).

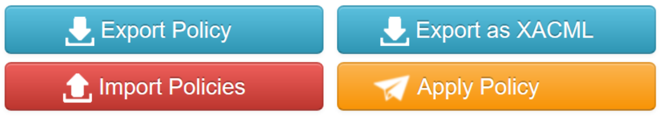
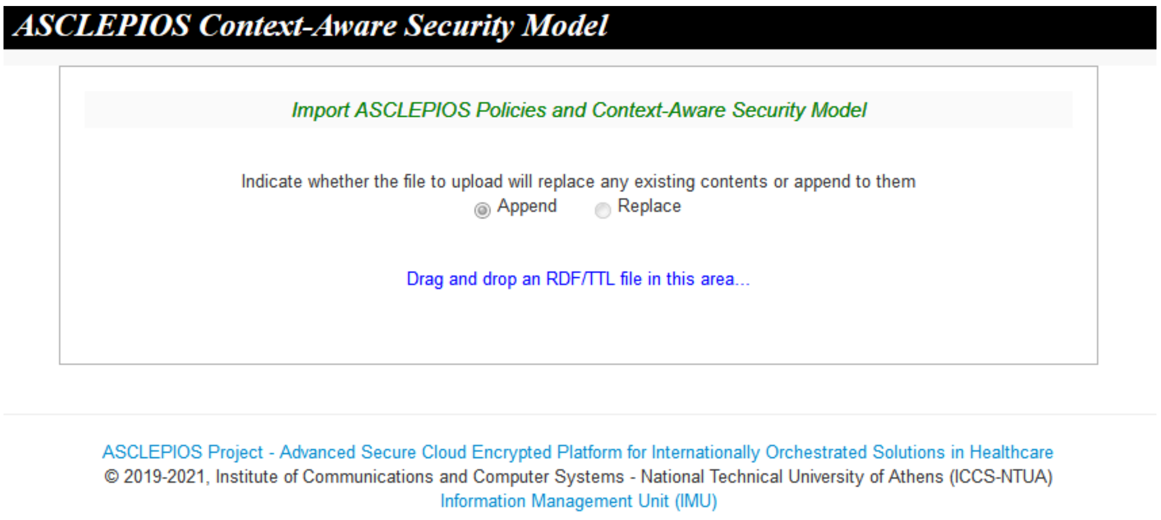


Figure 44 – Import/Export ABAC Policy editor buttons

In order to import an ABAC Policy into AMPLE, one can either use the *Import Policies* button in ABAC Policies Editor or the *Import Models* menu item in AMPLE menu. In both cases the same Import page loads (Figure 45) where the user can drag and drop the TTL policy file. Caution must be exercised to not replace pre-existing contents; i.e. Append import mode must be selected.



ASCLEPIOS Context-Aware Security Model

Import ASCLEPIOS Policies and Context-Aware Security Model

Indicate whether the file to upload will replace any existing contents or append to them
[✓ Append] Replace

Drag and drop an RDF/TTL file in this area...

ASCLEPIOS Project - Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare
© 2019-2021, Institute of Communications and Computer Systems - National Technical University of Athens (ICCS-NTUA)
Information Management Unit (IMU)

Figure 45 – Import page

In order to export a (single) ABAC Policy the user can click on *Export Policy* button in ABAC Policies editor. A TTL file containing the selected policy will be downloaded.

A second option is to export the selected ABAC Policy as an XACML file, which can be used in any XACML-capable application. The user needs to click on *Export as XACML* button in ABAC Policies editor, and an XACML file with the policy will be downloaded. It must be noted that XACML files cannot be imported back to AMPLE.

Interpret and Submit an ABAC Policy

ABAC Policies editor provide the capability to convert the selected policy into XACML and send it to ASCLEPIOS ABAC policy enforcement mechanism in order to put it into effect immediately. This can be achieved by pressing the *Apply Policy* button, next to Import/Export buttons (Figure 44).

I.3. ABAC Policies Editor REST API

Apart from a graphical user interface, ABAC Policies Editor provides a REST API in order to enable external clients to use its functionality through REST calls. The relevant endpoints, operations and message formats are detailed in the following Table 6, Table 7 and Table 8.

Table 6 – REST API of ABAC Policies Editor – Policies part

REST endpoint and Verb	Request	Response

getTopLevelPolicies
`GET /opt/abac-policies/`
 	
Description: Returns an array with all ABAC policies. Their full descriptions are contained in the array, as shown in the example at the Response column.
	`n/a`	`application/json`
		`<AbacPolicy> array`
		
		`[`
		`  {`
		`    "id": "string",`
		`    "name": "string",`
		`    "uri": "string",`
		`    "type": "ABAC-POLICY",`
		`    "description": "string",`
		`    "createTimestamp":`
		`      "2020-03-09T07:51:38.089",`
		`    "lastUpdateTimestamp":`
		`      "2020-03-09T07:51:38.089",`
		`    "policyCombiningAlgorithm":`
		`      "string"`
		`  }`
		`]`

getAllPolicies
`GET /opt/abac-policies/all`
 	
Description: Same as `getTopLevelPolicies`.
	`n/a`	`application/json`
		`<AbacPolicy> array`
		
		For an example see at **getTopLevelPolicies**

REST endpoint and Verb	Request	Response

getPolicy
`GET /opt/abac-policies/{policy_id}`
`policy_id: String`
 	
Description: Returns the description of the ABAC policy, specified by `policy_id` parameter, which matches to the `id` property of the policy.
	n/a	application/json
		AbacPolicy
		
		`{`
		`  "id": "string",`
		`  "name": "string",`
		`  "uri": "string",`
		`  "type": "ABAC-POLICY",`
		`  "description": "string",`
		`  "createTimestamp":`
		`    "2020-03-09T07:51:38.089",`
		`  "lastUpdateTimestamp":`
		`    "2020-03-09T07:51:38.089",`
		`  "policyCombiningAlgorithm":`
		`    "string"`
		`}`

REST endpoint and Verb	Request	Response

createPolicy
`PUT /opt/abac-policies/`
 	
Description: Creates a new ABAC policy with no rules. It requires the full definition of the new policy in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbacPolicy
	
	`{`
	`  "id": "string",`
	`  "name": "string",`
	`  "uri": "string",`
	`  "type": "ABAC-POLICY",`
	`  "description": "string",`
	`  "createTimestamp":`
	`    "2020-03-09T07:51:38.089",`
	`  "lastUpdateTimestamp":`
	`    "2020-03-09T07:51:38.089",`
	`  "policyCombiningAlgorithm":`
	`    "string"`
	`}`
		text/plain
		string

updatePolicy
`POST /opt/abac-policies/{policy_id}`
`policy_id: String`
 	
Description: Replaces the definition of an existing ABAC policy, specified by `policy_id` parameter, with a new definition. It requires the full (new) definition in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbacPolicy
	
	See at **createPolicy**
		text/plain
		string

REST endpoint and Verb	Request	Response

deletePolicy
`DELETE /opt/abac-policies/{policy_id}`
`policy_id: String`
 	
Description: Deletes an existing ABAC policy, specified by `policy_id` parameter. Policy must not have any rules, otherwise the operation fails.
	`n/a`	text/plain
		`string`

deletePolicyAndRules
`DELETE /opt/abac-policies/{policy_id}/all`
`policy_id: String`
 	
Description: Deletes an existing ABAC policy, specified by `policy_id` parameter, and all its rules.
	`n/a`	text/plain
		`string`

REST endpoint and Verb	Request	Response

getPolicyRules
`GET /opt/abac-policies/{policy_id}/rules`
`policy_id: String`
 	
Description: Returns an array with all ABAC policy rules of the policy specified by `policy_id` parameter, which matches to the id property of the attribute.
	n/a	application/json
		`<AbacRule> array`
		
		`[`
		`  {`
		`    "id": "string",`
		`    "name": "string",`
		`    "uri": "string",`
		`    "type": "ABAC-RULE",`
		`    "description": "string",`
		`    "createTimestamp":`
		`      "2020-03-09T07:52:27.331",`
		`    "lastUpdateTimestamp":`
		`      "2020-03-09T07:52:27.331",`
		`    "rulePolicy": {`
		`      "id": "string",`
		`      "uri": "string",`
		`      "type": "ABAC-POLICY",`
		`      "name": "string",`
		`      "description": "string",`
		`      "createTimestamp":`
		`        "2020-03-09T07:52:32.455",`
		`      "lastUpdateTimestamp":`
		`        "2020-03-09T07:52:32.455",`
		`      "policyCombiningAlgorithm":`
		`        "string"`
		`    },`
		`    "ruleOutcome": "string",`
		`    "ruleExpression": {}`
		`  }`
		`]`

Table 7 – REST API of ABAC Policies Editor – Rules part

REST endpoint and Verb	Request	Response

getRule
`GET /opt/abac-policies/rule/{rule_id}`
`rule_id: String`
 	
Description: Returns the description of the ABAC policy rule, specified by `rule_id` parameter, which matches to the `id` property of the rule.
	n/a	application/json
		<AbacRule> array
		
		`{`
		`  "id": "string",`
		`  "name": "string",`
		`  "uri": "string",`
		`  "type": "ABAC-RULE",`
		`  "description": "string",`
		`  "createTimestamp":`
		`    "2020-03-09T07:52:27.331",`
		`  "lastUpdateTimestamp":`
		`    "2020-03-09T07:52:27.331",`
		`  "rulePolicy": {`
		`    "id": "string",`
		`    "uri": "string",`
		`    "type": "ABAC-POLICY",`
		`    "name": "string",`
		`    "description": "string",`
		`    "createTimestamp":`
		`      "2020-03-09T07:52:32.455",`
		`    "lastUpdateTimestamp":`
		`      "2020-03-09T07:52:32.455",`
		`    "policyCombiningAlgorithm":`
		`      "string"`
		`  },`
		`  "ruleOutcome": "string",`
		`  "ruleExpression": {}`
		`}`

createRule
`PUT /opt/abac-policies/rule/`
 	
Description: Creates a new ABAC policy rule. It requires the full definition of the new rule in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbacRule
	
	`{`
	`  "id": "string",`
	`  "name": "string",`
	`  "uri": "string",`
	`  "type": "ABAC-RULE",`
	`  "description": "string",`
	`  "createTimestamp":`
	`    "2020-03-09T07:52:27.331",`
	`  "lastUpdateTimestamp":`
	`    "2020-03-09T07:52:27.331",`
	`  "rulePolicy": {`
	`    "id": "string",`
	`    "uri": "string",`
	`    "type": "ABAC-POLICY",`
	`    "name": "string",`
	`    "description": "string",`
	`    "createTimestamp":`
	`      "2020-03-09T07:52:32.455",`
	`    "lastUpdateTimestamp":`
	`      "2020-03-09T07:52:32.455",`
	`    "policyCombiningAlgorithm":`
	`      "string"`
	`  },`
	`  "ruleOutcome": "string",`
	`  "ruleExpression": {}`
	`}`
		text/plain
		string

updateRule
`POST /opt/abac-policies/rule/{rule_id}`
`rule_id: String`
 	
Description: Replaces the definition of an existing rule, specified by `rule_id` parameter, with a new definition. It requires the full (new) definition in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbacRule
	
	See at **createRule**
		text/plain
		string

deleteRule
`DELETE /opt/abac-policies/rule/{rule_id}`
`rule_id: String`
 	
Description: Deletes an existing rule, specified by the `rule_id` parameter.
	n/a	text/plain
		string

Table 8 – REST API of ABAC Policy-to-XACML interpreter

REST endpoint and Verb	Request	Response

abacPolicyToXacml
`GET /opt/interpreter/abac-policy-to-xacml/{policy_id}`
`policy_id: String`
 	
Description: Converts and returns the policy, specified by `policy_id` parameter, into XACML format.
	`n/a`	text/xacml+xml
		`string`
		
		`<?xml version="1.0" encoding="UTF-8"?>`
		`<xacml3:Policy`
		` xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"`
		` PolicyId="string"`
		` RuleCombiningAlgId="string"`
		` Version="1.0">`
		
		`  <xacml3:Description><![CDATA[`
		`    Policy Name: string`
		`    Description: string`
		`  ]]></xacml3:Description>`
		`  <!-- Rule Name:   string`
		`       Description: string`
		`  -->`
		`  <xacml3:Rule Effect="string"`
		`    RuleId="string">`
		`  </xacml3:Rule>`
		
		`</xacml3:Policy>`

I.4. ABAC Policies Serialization

ABAC Policies Editor stores ABAC Policy changes in Models Store as RDF triples. The following listing provides a sample ABAC Policy export in RDF/TTL format. The policy includes two rules with no condition.

Table 9 – Sample ABAC Policy export in RDF/TTL format

```
# Definition of ABAC policy: Second abac policy
<http://www.asclepios.eu/abac/ASCLEPIOS-ABAC-POLICY#144e8e20-2068-4cc2-ae77-
efe8acf06015>
    a
        <http://www.asclepios.eu/abac#ASCLEPIOS-ABAC-POLICY> ;
    <http://purl.org/dc/elements/1.1/type>
        "ABAC-POLICY" ;
    <http://purl.org/dc/terms/URI>
        "asclepios:144e8e20-2068-4cc2-ae77-efe8acf06015" ;
    <http://purl.org/dc/terms/description>
        "Second abac policy" ;
    <http://purl.org/dc/terms/identifier>
        "144e8e20-2068-4cc2-ae77-efe8acf06015" ;
    <http://purl.org/dc/terms/modified>
        "2020-03-09T09:52:32.455+02:00"                            ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/title>
        "ABAC Policy #2" ;
    <http://www.asclepios.eu/abac/ABAC-POLICY#combining-algorithm>
     "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable";
    <http://www.asclepios.eu/casm/types#class>
        "eu.asclepios.ample.model.abac.AbacPolicy" .


# Definition of ABAC rule: Rule 2-a
<http://www.asclepios.eu/abac/ASCLEPIOS-ABAC-RULE#8e05782c-e559-420b-b428-
a7851f804c91>
    a
        <http://www.asclepios.eu/abac#ASCLEPIOS-ABAC-RULE> ;
    <http://purl.org/dc/elements/1.1/type>
        "ABAC-RULE" ;
    <http://purl.org/dc/terms/URI>
        "asclepios:8e05782c-e559-420b-b428-a7851f804c91" ;
    <http://purl.org/dc/terms/created>
        "2020-03-09T09:52:27.331+02:00"                             ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/identifier>
        "8e05782c-e559-420b-b428-a7851f804c91" ;
    <http://purl.org/dc/terms/modified>
        "2020-03-09T09:52:27.331+02:00"                             ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/title>
        "Rule 2-a" ;
    <http://www.asclepios.eu/abac/ABAC-RULE#outcome>
        "PERMIT" ;
    <http://www.asclepios.eu/casm/types#class>
        "eu.asclepios.ample.model.abac.AbacRule" ;
    <http://www.paasword.eu/ABAC-RULE#policy>
        <http://www.asclepios.eu/abac/ASCLEPIOS-ABAC-POLICY#144e8e20-2068- ↙
                                                    4cc2-ae77-efe8acf06015> .


# Definition of ABAC rule: Rule 2-b
<http://www.asclepios.eu/abac/ASCLEPIOS-ABAC-RULE#ae12b9be-c569-4599-a0fd-
cfbf224788ce>
    a
        <http://www.asclepios.eu/abac#ASCLEPIOS-ABAC-RULE> ;
    <http://purl.org/dc/elements/1.1/type>
        "ABAC-RULE" ;
    <http://purl.org/dc/terms/URI>
        "asclepios:ae12b9be-c569-4599-a0fd-cfbf224788ce" ;
    <http://purl.org/dc/terms/created>
        "2020-03-09T09:52:57.022+02:00"                                 ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/identifier>
        "ae12b9be-c569-4599-a0fd-cfbf224788ce" ;
    <http://purl.org/dc/terms/modified>
        "2020-03-09T09:52:57.022+02:00"                                 ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/title>
        "Rule 2-b" ;
    <http://www.asclepios.eu/abac/ABAC-RULE#outcome>
        "DENY" ;
    <http://www.asclepios.eu/casm/types#class>
        "eu.asclepios.ample.model.abac.AbacRule" ;
    <http://www.paasword.eu/ABAC-RULE#policy>
        <http://www.asclepios.eu/abac/ASCLEPIOS-ABAC-POLICY#144e8e20-2068- ↙
                                               4cc2-ae77-efe8acf06015> .
```

I.5. ABAC Policies Interpretation

ABAC Policies Editor invokes the ABAC Policy-to-XACML Interpreter component of AMPLE in order to convert the selected ABAC Policy into XACML, which can either be downloaded as a file or submitted to the ABAC Policies Enforcement Mechanism for applying it. The following listing provides a sample ABAC Policy export as XACML.

Table 10 – Sample ABAC Policy as XACML

```
<?xml version="1.0" encoding="UTF-8"?>

<xacml3:Policy xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  PolicyId="144e8e20-2068-4cc2-ae77-efe8acf06015"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-          ↙
                                             algorithm:first-applicable"
  Version="1.0">

    <xacml3:Description><![CDATA[
        Policy Name: ABAC Policy #2
        Description: Second abac policy
    ]]></xacml3:Description>

    <!-- Rule Name:   Rule 1-a
         Description:
    -->
    <xacml3:Rule Effect="PERMIT" RuleId="8e05782c-e559-420b-b428-a7851f804c91">
    </xacml3:Rule>

    <!-- Rule Name:   Rule 1-b
         Description:
    -->
    <xacml3:Rule Effect="DENY" RuleId="ae12b9be-c569-4599-a0fd-cfbf224788ce">
    </xacml3:Rule>

</xacml3:Policy>
```

5 Attribute-Based Encryption Policies Editor

In this chapter the *Attribute-Based Encryption Policies Editor* is presented and detailed.

ABE Policies resemble to Boolean expressions of (a subset of) CASM attributes, used for creating key pairs for encrypting/decrypting protected resources, according to the ABE method described in deliverable D1.2 [3]. The key pair is generated based on specific attribute values and the decryption key is provided to the eligible users (i.e. those who must be granted access to the protected resources). ABE policies are evaluated (or applied), at resource encryption time as well as every time an access is attempted to a protected resource. When that happens, the whole policy expression is evaluated.

An ABE Policy expression can either be a composite or a simple Boolean expression. A composite expression comprises of sub-expressions, combined using a Boolean operator (AND/OR). A simple expression is typically a comparison expression between an attribute (acting as a value placeholder) and a constant value. Additionally, the *k-of-N* operator is also supported (evaluates to true when at least k sub-expressions evaluate to true).

In the context of AMPLE, ABE Policies are uniquely identified with an *Id* property, titled with a *Name*, and they can optionally have a *Description*. They can also be referenced using a Universal Resource Identifier (*URI*).

In the following sections the ABE Policies editor is presented in detail.

I.1. Usage Scenarios

The main goal of ABE Policies Editor is to create and manage the ABE Policies of ASCLEPIOS platform. This goal can be broken down to a series of specific capabilities that must be offered to the user for creating, updating, deleting and retrieving the ABE policies, as well as for importing and exporting them to files. A full list of ABE Policies Editor capabilities (and related usage scenarios) is given next.

- List existing ABE Policies
- Retrieve an ABE Policy's details
- Create a new ABE Policy
- Modify or Delete an existing ABE Policy
- Import an ABE Policy from a file
- Export an ABE Policy to a file
- Submit an ABE Policy to ABE Service for use (i.e. apply ABE policy)

The ABE Policies retrieval and editing capabilities are depicted in Figure 46, whereas the import/export, interpretation and submission to ABE Service capabilities are depicted in Figure 47.

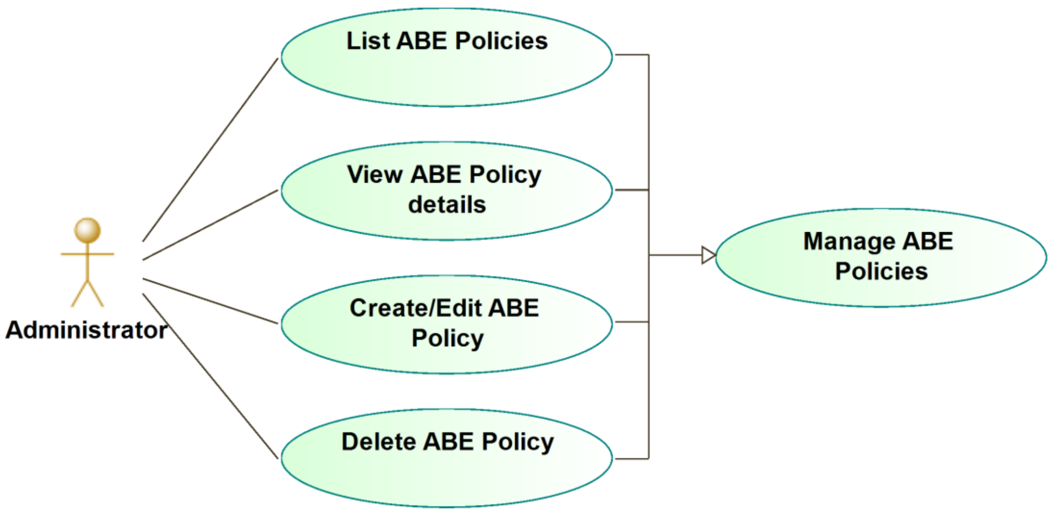


Figure 46 – ABE Policies management use case

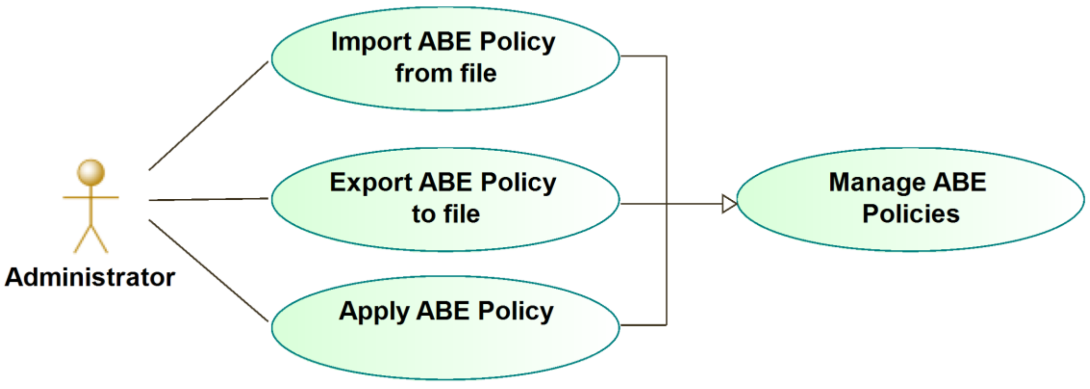


Figure 47 – ABE Policies Import/Export, and submission to ABE Service use case

I.2. Walkthrough

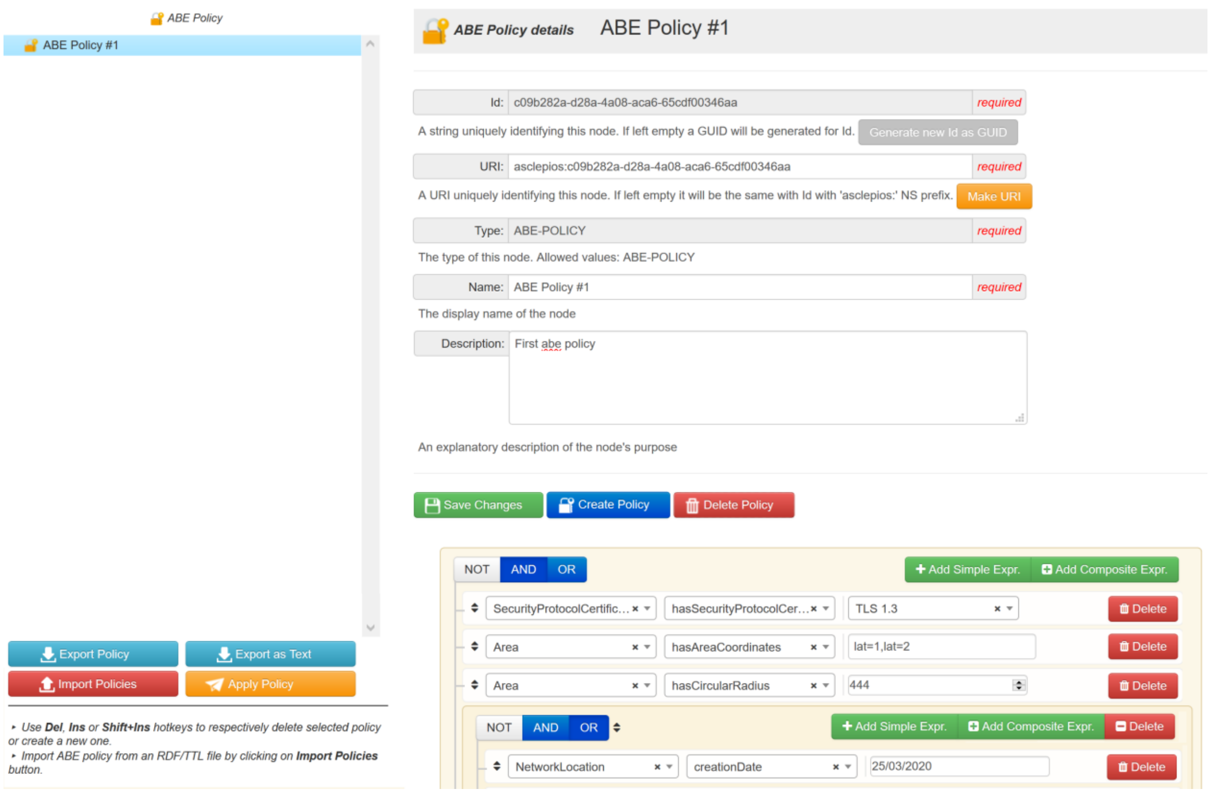
In this section the graphical user interface of the ABE Policies Editor is briefly presented along with a short walkthrough of its operation. The editor can be accessed through the AMPLE web page.

ABE Policies Editor

ABE Policies Editor opens by pressing the button named *"ABE Policies"* in the Welcome screen or the respective button in menu. Figure 48 gives an overview of the ABE Policies Editor.

The page is vertically divided in two notable regions. The left-hand part of the page contains a list of all existing ABE Policies. Clicking on an ABE Policy loads its details into the details form in the right-hand side of the page. Right clicking on an item in the list will open the context menu which offers actions related to the selected item.

The details form (in the right-hand side of the page) encompasses fields pertaining to ABE Policies definition. Under the details form a row of buttons exists. These buttons can be used to save any changes made in details form, create new ABE Policies or delete the currently displayed item in details form. Buttons get dimmed when the corresponding operation is not available in a particular case (e.g. when creating a policy, all buttons but *Save* are dimmed).



ABE Policy details ABE Policy #1

Id:	c09b282a-d28a-4a08-aca6-65cdf00346aa	*required*
URI:	asclepios:c09b282a-d28a-4a08-aca6-65cdf00346aa	*required*
Type:	ABE-POLICY	*required*
Name:	ABE Policy #1	*required*
Description:	First abe policy	

A string uniquely identifying this node. If left empty a GUID will be generated for Id. Generate new Id as GUID

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix. Make URI

The type of this node. Allowed values: ABE-POLICY

The display name of the node

An explanatory description of the node's purpose

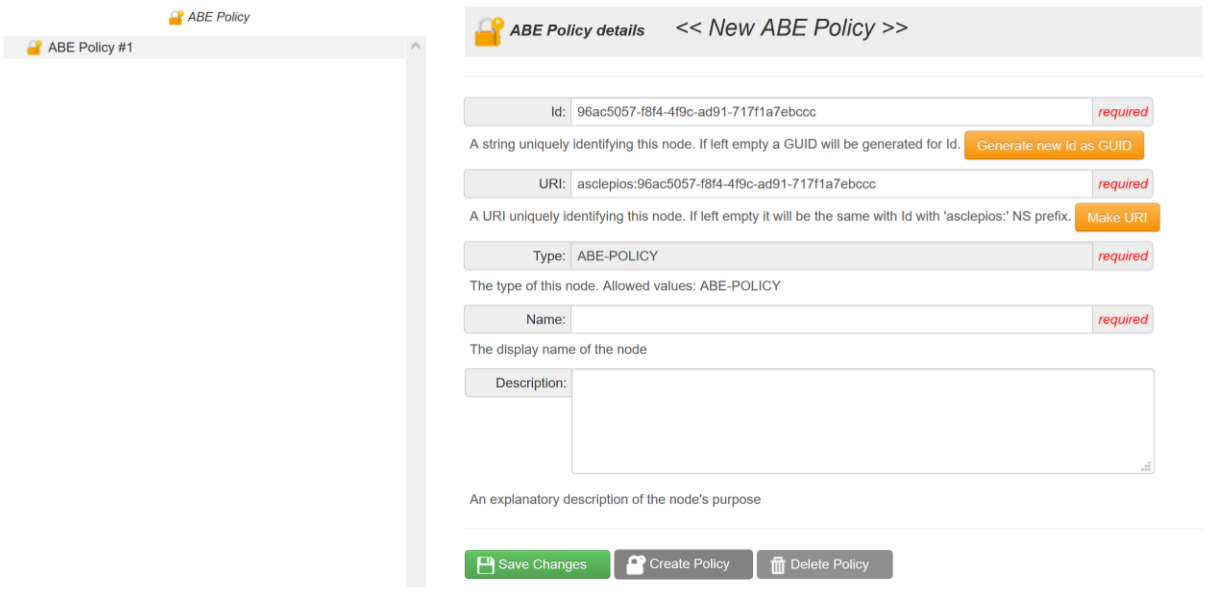
SecurityProtocolCertific…	hasSecurityProtocolCer…	TLS 1.3
Area	hasAreaCoordinates	lat=1,lat=2
Area	hasCircularRadius	444
NetworkLocation	creationDate	25/03/2020

- *Use **Del**, **Ins** or **Shift+Ins** hotkeys to respectively delete selected policy or create a new one.*
- *Import ABE policy from an RDF/TTL file by clicking on **Import Policies** button.*

Figure 48 – ABE Policies Editor page

Create new ABE Policy

A user can create a new ABE Policy by clicking on the *"Create Policy"* button under the details form (see Figure 49). This action will clear details form, and prefill *Id*, *Type* and *URI* fields with automatically created values values. The user has the opportunity to modify these values before saving them (Figure 50). When ready, the user can click on *Save* button (colored green) to submit information to the server for storing.



ABE Policy details << New ABE Policy >>

Id:	96ac5057-f8f4-4f9c-ad91-717f1a7ebccc	*required*
URI:	asclepios:96ac5057-f8f4-4f9c-ad91-717f1a7ebccc	*required*
Type:	ABE-POLICY	*required*
Name:		*required*
Description:		

A string uniquely identifying this node. If left empty a GUID will be generated for Id. Generate new Id as GUID

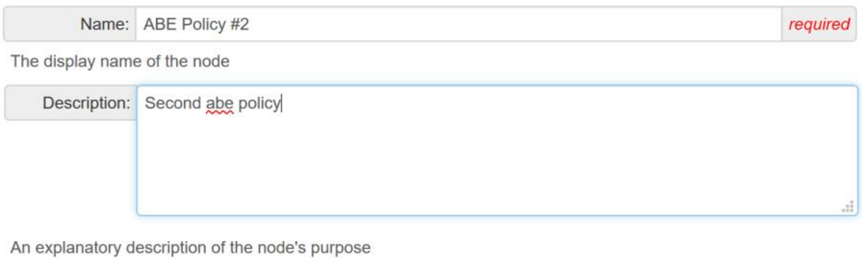
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix. Make URI

The type of this node. Allowed values: ABE-POLICY

The display name of the node

An explanatory description of the node's purpose

Figure 49 – New ABE Policy form



Name: ABE Policy #2 *required*

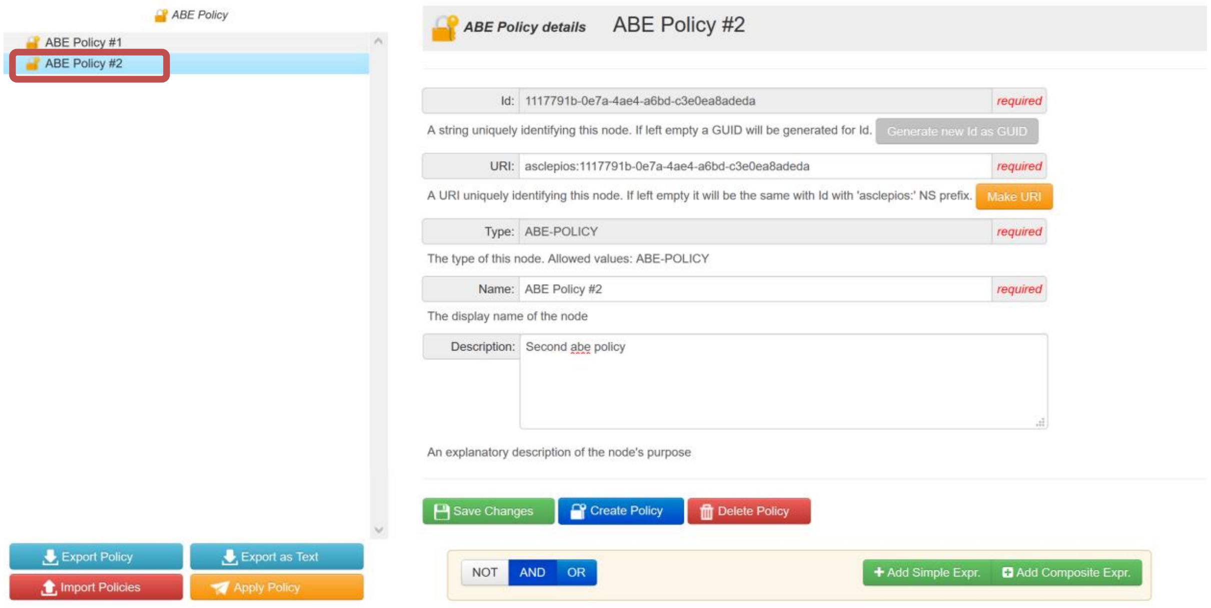
The display name of the node

Description: Second abe policy

An explanatory description of the node's purpose

Figure 50 – Filling in new ABE Policy details

After saving changes, the ABE Policies list on the left-hand side of the page will refresh in order to reflect the changes, i.e. including the new ABE Policy (Figure 51).



ABE Policy

ABE Policy #1
ABE Policy #2

ABE Policy details ABE Policy #2

Id: 1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda *required*

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda *required*

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: ABE-POLICY *required*

The type of this node. Allowed values: ABE-POLICY

Name: ABE Policy #2 *required*

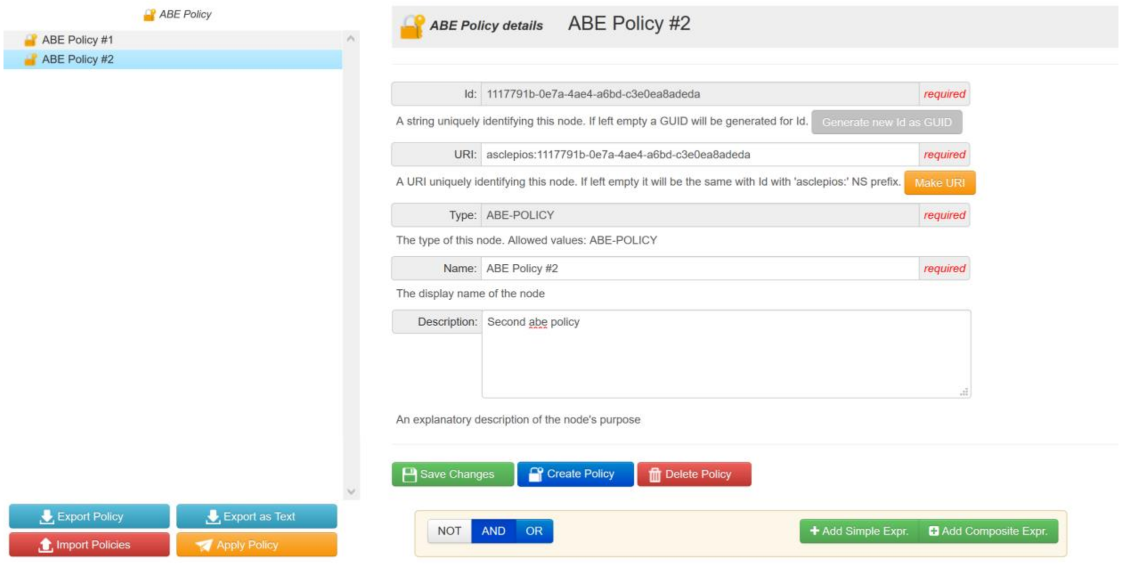
The display name of the node

Description: Second abe policy

An explanatory description of the node's purpose

Figure 51 – New ABE Policy

An integral part of an ABE Policy is its *Expression*, which is a boolean expression of attributes and constant values, as they have been specified in CASM. The expression controls the generation of the resource encryption/decryption keys.



ABE Policy

ABE Policy #1
ABE Policy #2

ABE Policy details ABE Policy #2

Id: 1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda *required*

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

URI: asclepios:1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda *required*

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

Type: ABE-POLICY *required*

The type of this node. Allowed values: ABE-POLICY

Name: ABE Policy #2 *required*

The display name of the node

Description: Second abe policy

An explanatory description of the node's purpose

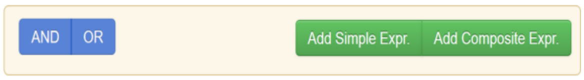
Figure 52 – ABE Policy expression editor

The graphical expression editor, which resides under the details form, can be used to create the policy expression (Figure 52). The expression editor provides dynamic lists (as combo boxes) for easier search and selection of the required CASM attributes. Expression is saved along with the details form data, when the *Save* button is pressed.

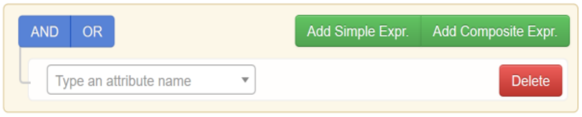
Initially the expression is empty. Pressing the *Add Simple Expr.* button (on the right hand side) a new expression is started containing a simple clause. More clauses can be added in the following lines by pressing the same button. Each line contains the definition of a simple expression clause of the form: **`<Attribute> <Operator> <Value>`**
For example: `SeurityProtocolCertificate = "TLS"`

Clauses are combined using the boolean operator indicated by the selected button on the to top left corner of the expression builder (AND/OR). Subexpressions can be added by pressing the *Add Composite Expr.* button. Subexpressions have their own boolean operator and simple clauses. The following screenshots give an impression of expression builder (Figure 53).

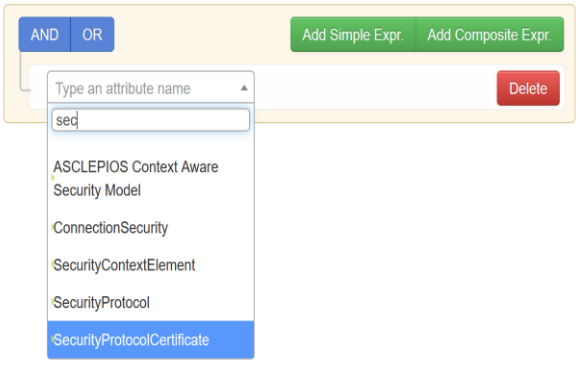
Empty expression (default)



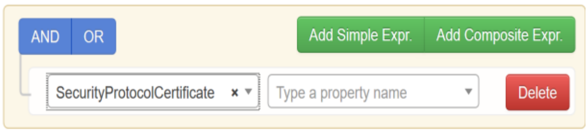
Pressing "Add Simple Expr." a new simple expression is added



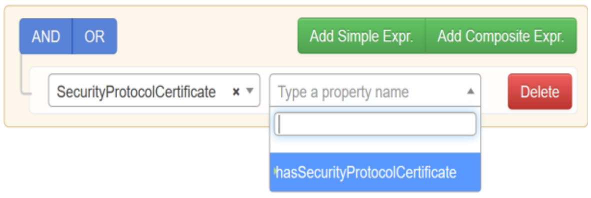
Using selection box an attribute can be selected



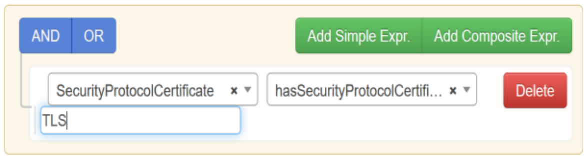
Properties selection box appears



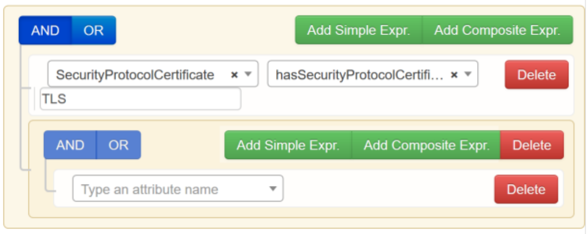
Using second selection box an attribute's property can be selected



Property value input appears



Pressing "Add Composite Expr." a sub-expression is added



Example expression with sub-expression

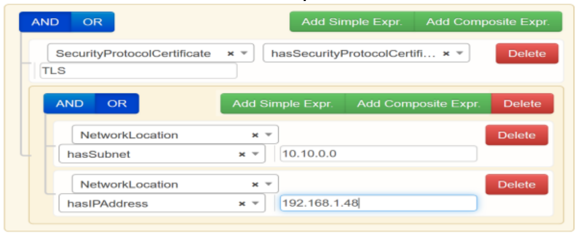
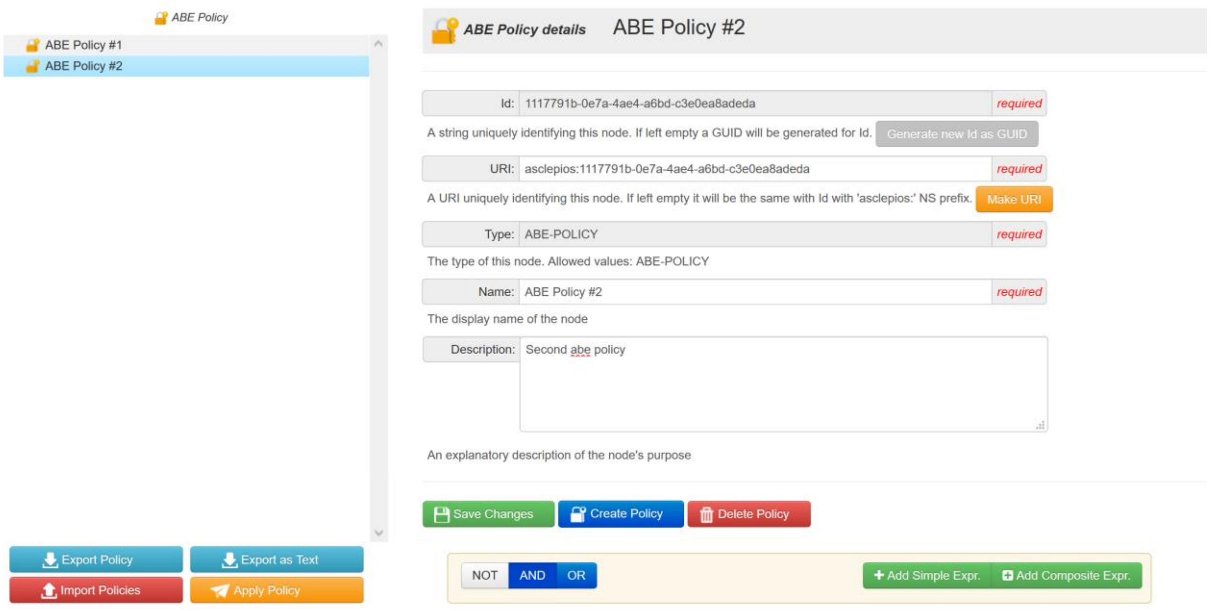


Figure 53 – ABE Policy expression editing with expression builder

Edit an existing ABE Policy

Selecting an ABE Policy in the list on the left will load its information in the details form on the right (see Figure 54). The user has the opportunity to modify the information in the form (except *Id* and *Type*) (see Figure 55).



ABE Policy details ABE Policy #2

Id:	1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda	required
URI:	asclepios:1117791b-0e7a-4ae4-a6bd-c3e0ea8adeda	required
Type:	ABE-POLICY	required
Name:	ABE Policy #2	required
Description:	Second abe policy	

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

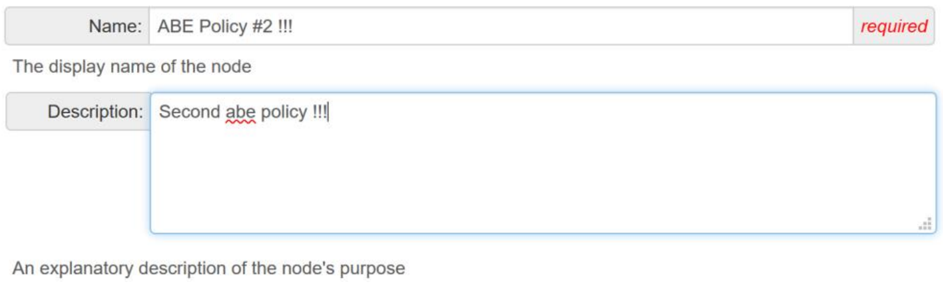
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

The type of this node. Allowed values: ABE-POLICY

The display name of the node

An explanatory description of the node's purpose

Figure 54 – Edit an existing ABE Policy



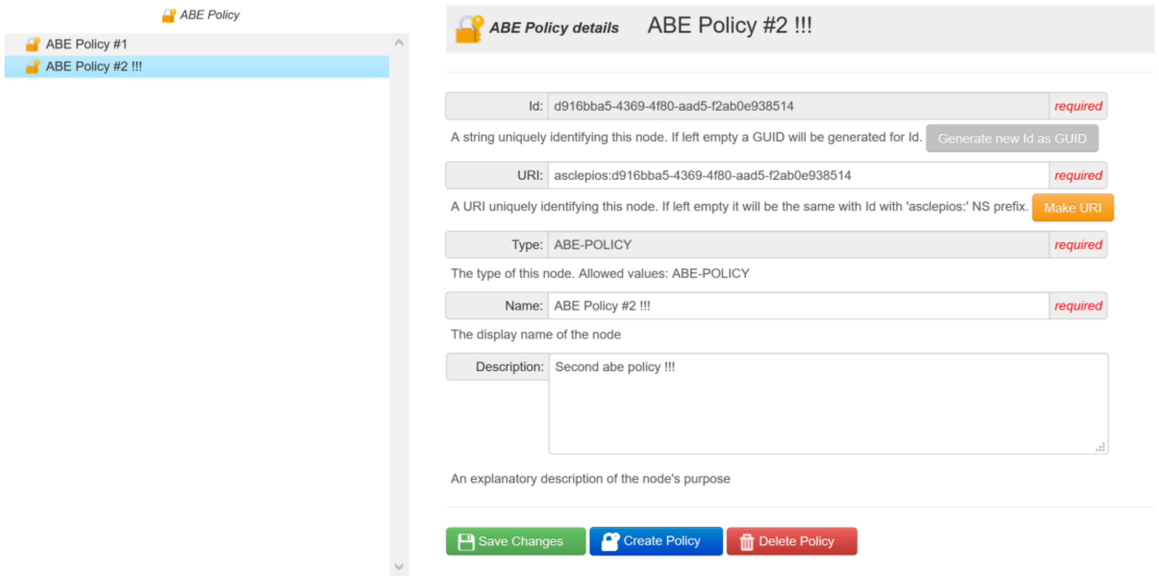
Name:	ABE Policy #2 !!!	required
Description:	Second abe policy !!!	

The display name of the node

An explanatory description of the node's purpose

Figure 55 – Change an ABE Policy's details

After saving changes, the list on the left-hand side of the page will refresh in order to reflect any changes (e.g. displaying the new ABE Policy name) (see Figure 56).



ABE Policy details ABE Policy #2 !!!

Id:	d916bba5-4369-4f80-aad5-f2ab0e938514	required
URI:	asclepios:d916bba5-4369-4f80-aad5-f2ab0e938514	required
Type:	ABE-POLICY	required
Name:	ABE Policy #2 !!!	required
Description:	Second abe policy !!!	

A string uniquely identifying this node. If left empty a GUID will be generated for Id.

A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.

The type of this node. Allowed values: ABE-POLICY

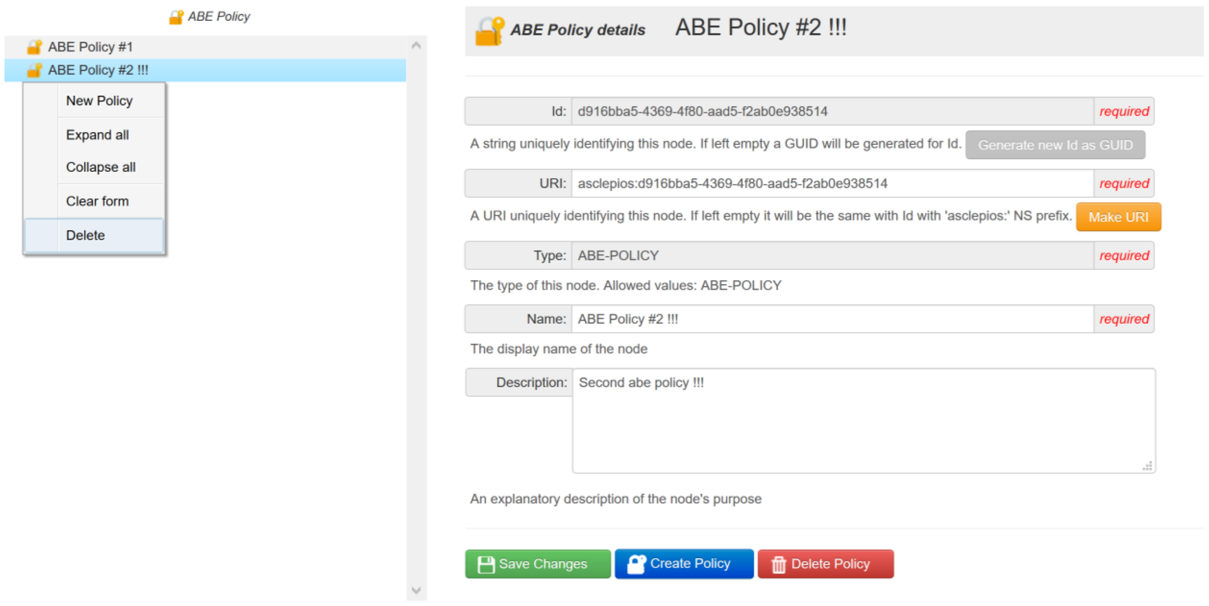
The display name of the node

An explanatory description of the node's purpose

Figure 56 – Changed ABE Policy in the list

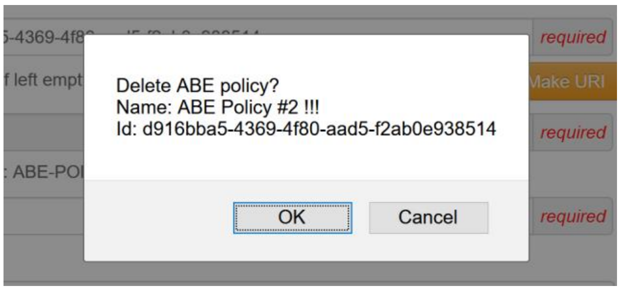
Delete an ABE Policy

Selecting an ABE Policy in the list on the left will load its details in the details form on the right. User has the opportunity to delete that particular ABE Policy. This can be achieved by pressing the *Delete* button residing under details form or using the corresponding option of the context menu, accessible with right-click (see Figure 57). In both cases the user will need to confirm operation before deletion is carried out (see Figure 58).



Id: d916bba5-4369-4f80-aad5-f2ab0e938514 *required*
A string uniquely identifying this node. If left empty a GUID will be generated for Id.
URI: asclepios:d916bba5-4369-4f80-aad5-f2ab0e938514 *required*
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.
Type: ABE-POLICY *required*
The type of this node. Allowed values: ABE-POLICY
Name: ABE Policy #2 !!! *required*
The display name of the node
Description: Second abe policy !!!
An explanatory description of the node's purpose

Figure 57 – Deleting an ABE Policy using context menu



Delete ABE policy?
Name: ABE Policy #2 !!!
Id: d916bba5-4369-4f80-aad5-f2ab0e938514

Figure 58 – Confirm ABE Policy deletion

Import/Export an ABE Policy

It is possible to import and export ABE policies as RDF/TTL files for backup or migration purposes (Figure 59).

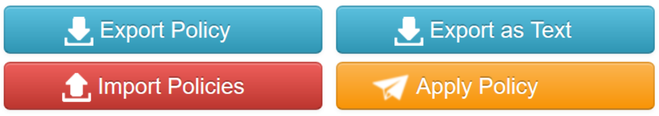
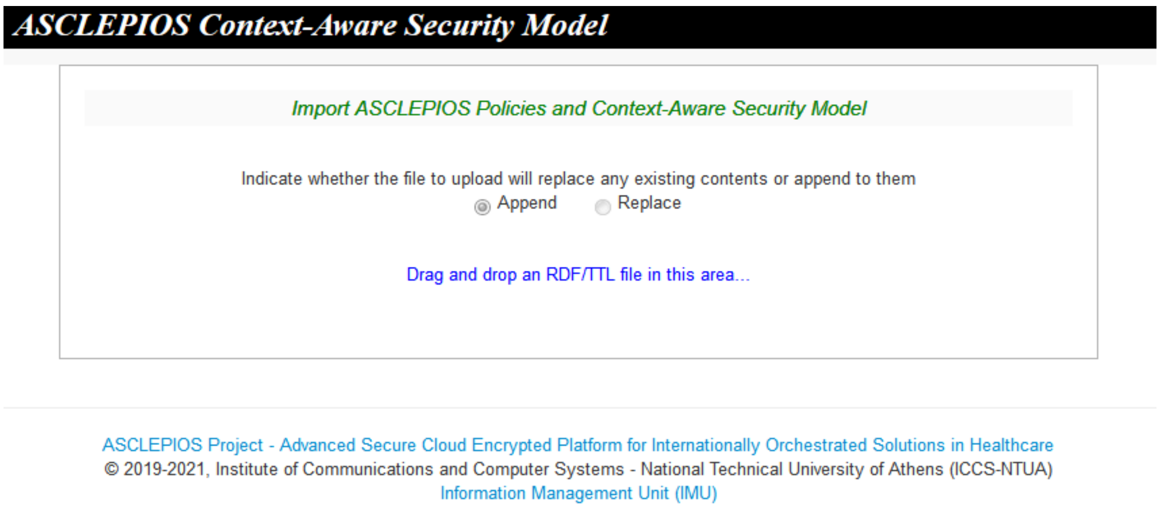


Figure 59 – Import/Export ABE Policy editor buttons

In order to import an ABE Policy into AMPLE, one can either use the *Import Policies* button in ABE Policies Editor or the *Import Models* menu item in AMPLE menu. In both cases the same *Import page* loads (Figure 60), where the user can drag and drop the TTL policy file. Caution must be exercised to not replace pre-existing contents; i.e. Append import mode must be selected.


ASCLEPIOS Context-Aware Security Model

Import ASCLEPIOS Policies and Context-Aware Security Model

Indicate whether the file to upload will replace any existing contents or append to them
[✓ Append] Replace

Drag and drop an RDF/TTL file in this area...

ASCLEPIOS Project - Advanced Secure Cloud Encrypted Platform for Internationally Orchestrated Solutions in Healthcare
© 2019-2021, Institute of Communications and Computer Systems - National Technical University of Athens (ICCS-NTUA)
Information Management Unit (IMU)

Figure 60 – Import page

In order to export a (single) ABE Policy the user can click on *Export Policy* button in ABE Policies editor. A TTL file containing the selected policy will be downloaded.

A second option is to export the selected ABE Policy in the format used by the ABE Service. The user needs to click on Export as Text button in ABE Policies editor, and a text file with the policy in the appropriate format, will be downloaded. It must be noted that text files cannot be imported back to AMPLE.

Interpret and Submit an ABE Policy

ABE Policies editor provide the capability to convert the selected policy into the ABE Service format and send it to ABE Service in order to put it into effect immediately. This can be achieved by pressing the *Apply Policy* button, next to Import/Export buttons (Figure 59).

I.3. ABE Policies Editor REST API

Apart from a graphical user interface, ABE Policies Editor provides a REST API in order to enable external clients to use its functionality through REST calls. The relevant endpoints, operations and message formats are detailed in the following Table 11 and Table 12.

Table 11 – REST API of ABE Policies Editor

REST endpoint and Verb	Request	Response

getTopLevelPolicies
`GET /opt/abe-policies/`
 	
Description: Returns an array with all ABE policies. Their full descriptions are contained in the array, as shown in the example at the Response column.
	`n/a`	`application/json`
		`<AbePolicy> array`
		
		`[`
		`  {`
		`    "id": "string",`
		`    "name": "string",`
		`    "uri": "string",`
		`    "type": "ABE-POLICY",`
		`    "description": "string",`
		`    "createTimestamp":`
		`      "2020-03-16T20:39:27.176",`
		`    "lastUpdateTimestamp":`
		`      "2020-03-16T20:39:27.176",`
		`  }`
		`]`

getAllPolicies
`GET /opt/abe-policies/all`
 	
Description: Same as `getTopLevelPolicies`.
	`n/a`	`application/json`
		`<AbePolicy> array`
		
		For an example see at **getTopLevelPolicies**

REST endpoint and Verb	Request	Response

getPolicy
`GET /opt/abe-policies/{policy_id}`
`policy_id: String`
 	
Description: Returns the description of the ABE policy, specified by `policy_id` parameter, which matches to the `id` property of the policy.
	n/a	application/json
		AbePolicy
		
		`{`
		`  "id": "string",`
		`  "name": "string",`
		`  "uri": "string",`
		`  "type": "ABE-POLICY",`
		`  "description": "string",`
		`  "createTimestamp":`
		`    "2020-03-16T20:39:27.176",`
		`  "lastUpdateTimestamp":`
		`    "2020-03-16T20:39:27.176",`
		`  "policyCombiningAlgorithm":`
		`    "string"`
		`}`

createPolicy
`PUT /opt/abe-policies/`
 	
Description: Creates a new ABE policy. It requires the full definition of the new ABE policy in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbePolicy
	
	`{`
	`  "id": "string",`
	`  "name": "string",`
	`  "uri": "string",`
	`  "type": "ABE-POLICY",`
	`  "description": "string",`
	`  "createTimestamp":`
	`    "2020-03-16T20:39:27.176",`
	`  "lastUpdateTimestamp":`
	`    "2020-03-16T20:39:27.176",`
	`  "policyCombiningAlgorithm":`
	`    "string"`
	`}`
		text/plain
		string

REST endpoint and Verb	Request	Response

updatePolicy
`POST /opt/abe-policies/{policy_id}`
`policy_id: String`
 	
Description: Replaces the definition of an existing ABE policy, specified by `policy_id` parameter, with a new definition. It requires the full (new) policy definition in the Request body. It returns a plain text message with the result of the operation.
	application/json
	AbePolicy
	
	See at **createPolicy**
		text/plain
		string

deletePolicy
`DELETE /opt/abe-policies/{policy_id}`
`policy_id: String`
 	
Description: Deletes an existing ABE policy, specified by `policy_id` parameter.
	n/a	text/plain
		string

Table 12 – REST API of ABE Policy-to-ABE Service format interpreter

REST endpoint and Verb	Request	Response

abePolicyToText
`GET /opt/interpreter/abe-policy-to-text/{policy_id}`
`policy_id: String`
 	
Description: Converts the ABE policy, which is specified by `policy_id` parameter, into text format and sends it to the ABE Service.
	n/a	text/plain
		string
		
		`(SecurityProtocolCertificate='TLS' and (NetworkLocation_hasSubnet = '10.10.1.0/24' or PhysicalLocation_address = 'Building-1'))`

I.4. ABE Policies Serialization

ABE Policies Editor stores changes in Models Store as RDF triples. The following listing provides a sample ABE Policy export in RDF/TTL format.

Table 13 – Sample ABE Policy export in RDF/TTL format

```
# Definition of ABE policy: ABE policy #1
<http://www.asclepios.eu/abe/ASCLEPIOS-ABE-POLICY#73bb15cd-ad75-4465-a09c-
9af8a66abd81>
    a
        <http://www.asclepios.eu/abe#ASCLEPIOS-ABE-POLICY> ;
    <http://purl.org/dc/elements/1.1/type>
        "ABE-POLICY" ;
    <http://purl.org/dc/terms/URI>
        "asclepios:73bb15cd-ad75-4465-a09c-9af8a66abd81" ;
    <http://purl.org/dc/terms/description>
        "First abe policy" ;
    <http://purl.org/dc/terms/identifier>
        "73bb15cd-ad75-4465-a09c-9af8a66abd81" ;
    <http://purl.org/dc/terms/created>
        "2020-03-16T20:39:27.176"                                     ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/modified>
        "2020-03-16T20:39:27.176"                                     ↙
                               ^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/title>
        "ABE Policy #1" ;
    <http://www.asclepios.eu/casm/types#class>
        "eu.asclepios.ample.model.abe.AbePolicy" .
```

I.5. ABE Policies Interpretation

ABE Policies Editor invokes the ABE Policy-to-Text Interpreter component of AMPLE in order to convert the selected ABE Policy into a text format usable by ABE Service, which can either be downloaded as a file or submitted to ABE Service for applying it. The following listing provides a sample ABE Policy export in text format.

Table 14 – Sample ABE Policy in ABE Service text format

```
(SecurityProtocolCertificate = 'TLS' and
    (NetworkLocation_hasSubnet = '10.10.1.0/24' or
     PhysicalLocation_address = 'Building-1')
 )
```

6 Conclusions

To conclude, in this deliverable we reported on the development of all the appropriate editing mechanisms for updating the ASCLEPIOS context-aware security model and devising the necessary context-aware access policies. The first part of the editing functionalities refers to the ability to update or improve, according to the adopter's needs, the security context-aware model which is used as a common vocabulary for devising ASCLEPIOS access control policies. These access control policies will be enforced as part of two different authorisation paradigms that will be employed in sequence for achieving even higher levels of security controls. Therefore, separate editing functionalities are provided for creating ABAC and ABE policies that are enforced through the Asclepios Policy Enforcement mechanism and the ABE service.

During the next period these editors will continue to be improved with respect to user inteface enhancements or any other bug fixes on issues discovered during the integration work. Furthermore, a Policy Validator mechanism will be implemented and integrated in these editors, where policy developers will be able to define rules for checking policy correctness, completeness or for security awareness. Last, the findings during the security assessment process that have been provided in Appendix I - Asclepios Editor Technical Security Assessment , by Secura, will be considered for adding supplementary security enhancements in these editors.

7 References

1. Y., Verginadis et al., 2019. D3.1 ASCLEPIOS Security and Policies Model. ASCLEPIOS Deliverable
2. R., G., Roessink et al., 2020. D2.2 Attribute-Based Encryption, Dynamic Credentials and Ciphertext Delegation and Integration in Medical Devices. ASCLEPIOS Deliverable
3. A., Michalas et al., 2019. D1.2 ASCLEPIOS Reference Architecture, Security and E-health Use Cases, and Acceptance Criteria. ASCLEPIOS Deliverable

Appendix I - Asclepios Editor Technical Security Assessment (by Secura)

I.1. Assumptions

Inputs

1 **Database**

No assumptions need to be made for this component.

2 **Imported TTL file**

- The editor will only accept TTL files and will validate the content before uploading.
- The uploaded file will not be directly accessible after it has been uploaded.
- Users of this function will be trusted and the files uploaded are considered to be valid.

3 **Changes using model editor**

- Changes done on attributes follow a predefined template structure (type and format) and checks will be done before any change is applied.
- Only authorized users are able to make changes to attributes in the model editor.

Outputs

1 **Structured representation of attributes**

- If a person satisfies an attribute, they satisfy all sub-classes of that attribute.

2 **Definition of attributes**

- Data that is represented on screen will be properly filtered and sanitized to prevent the inclusion of malicious content.

3 **Exported TTL file**

No assumptions need to be made for this component.

I.2. Assumption analysis – potential threats

Inputs

1 **Database**

No potential threats were derived from the analysis of the Database component.

2 **Imported TTL file**

- A user that uploads a TTL file to be used in the editor can have executable content in the file. If the necessary checks are not performed, the file can cause harm to the rest of the editor users as arbitrary (and potentially harmful) code can be run on their machines.
- If a user uploads a TTL file and can use a URL to send that file to someone else, the recipient can be exposed to malicious content in the file, if such content is placed there by an attacker.

3 **Changes using model editor**

- A user can make changes to the editor that were not considered by the developers, such changes may result in undefined behaviors, enlarging the attack surface of the model editor.

Outputs

1 **Structured representation of attributes**

- A user can create a new attribute that is a top-level attribute, enabling them to satisfy all policies and therefore decrypt all data.

2 **Definition of attributes**

- A user can make changes to the policy editor such that when those changes are reflected to other users, arbitrary code can run putting their systems at risk.

3 **Exported TTL file**

No potential threats were derived from the TTL file export function.

I.3. Investigation

In order to investigate the model editor, the Web Application Security Testing methodology by OWASP was used. The process involved an active analysis of the Model Editor for weaknesses, technical flaws, or vulnerabilities. Due to the fact that there are no user-control related functionalities implemented in the editor and also the real environment that it will be deployed is not yet clear, it was decided that some of the sections from the testing methodology should be left out as they did not fit the purpose of this investigation.

The goal of the investigation was to assess possible vulnerabilities related to the individual components of the Model Editor. Specifically, in the scope of the assessment are the model editor model management component and the import models component. These items were individually assessed from a security point of view, as well as the interaction between them.

I.3.1. Transport Layer Security (TLS)

Communication with the application takes place entirely over a connection encrypted using TLS (HTTPS).
We observe that HTTP Strict Transport Security (HSTS) is used to enforce the use of TLS encryption. The Strict-Transport-Security header is present among the HTTPS headers returned by the server:

Client Request (HTTPS):

```
GET / HTTP/1.1
Host: localhost:9090
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Connection: close
Referer: https://localhost:9090/login
Cookie: JSESSIONID=2161EA394CD2E3B5316AA65231734AAF
Upgrade-Insecure-Requests: 1
```

Server Response:

```
HTTP/1.1 200
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Set-Cookie: JSESSIONID=56F94911EADA610B950E4EC4F9F81D80; Path=/; Secure; HttpOnly
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Content-Type: text/html;charset=UTF-8
Content-Language: en-US
Content-Length: 3149
```

The use of HSTS is important even if no HTTP service is active. An attacker who is able to perform a man-in-the-middle attack can replace https links of the application that are sent over an encrypted connection with http links, and then intercept the unencrypted traffic and forward it to the server in encrypted form.

Client request (HTTP):

```
GET / HTTP/1.1
Host: localhost:9090
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Connection: close
Upgrade-Insecure-Requests: 1
```

Server response:

```
Bad Request
This combination of host and port requires TLS.
```

The editor does not offer an HTTP service therefore all requests to that endpoint are rejected.

I.3.2. Cross-Site scripting

When data which is under control of a user is included in an (X)HTML document which is opened by the browser, it is important that characters with a special meaning (dependent on the location in the document) are escaped. If this escaping is not or incorrectly performed, this may have the consequence that an attacker is able to manipulate the structure of the (X)HTML document.

It may be possible to introduce JavaScript code, which is executed by the browser in the context of the vulnerable applications. This might lead attacker-introduced JavaScript to be executed in another user's browser.

Such vulnerabilities are known as Cross-Site Scripting (XSS). We distinguish two forms of XSS: stored and reflected.

With stored Cross-Site Scripting, the attacker can introduce JavaScript code. The code is stored in the application and is executed when a victim visits a page which includes the JavaScript code.

With reflected Cross-Site Scripting, the JavaScript code is only executed as a reaction to an action in the application through which the JavaScript code is entered. To abuse this, a victim must first (inadvertently) visit a URL which has been prepared by the attacker.

Testing for stored Cross-Site Scripting

Throughout the entire application, we have found no possibilities for stored Cross-Site Scripting. While information from user input is included in pages in various places, characters with a special meaning within (X)HTML are always properly escaped.

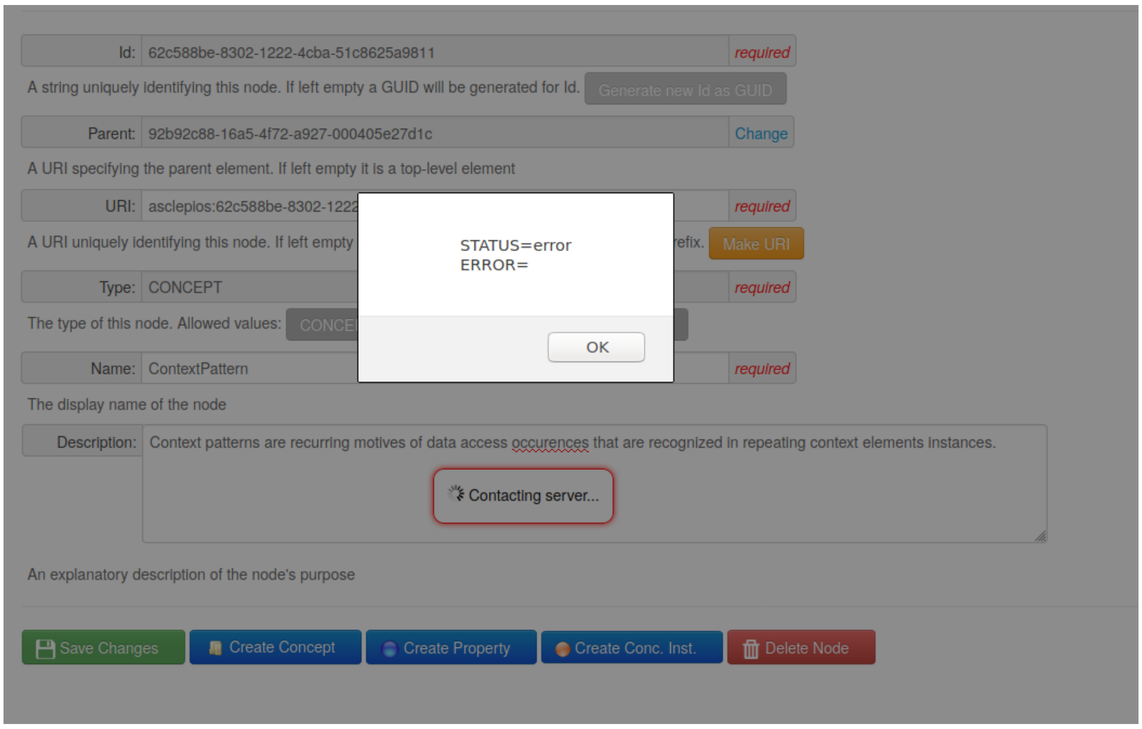
Testing for reflected Cross-site Scripting

The Schema Management form (https://localhost:9090/forms/admin/schema-mgnt.jsp) was tested for cross-site scripting vulnerabilities. The testing was done by placing a generic XSS testing string in the form's fields.

```
POST/gui/admin/save-attribute HTTP/1.1
Host: localhost:9090
[...]

{"id":"62c588be-8302-1222-4cba-51c8625a9811","uri":"asclepios:62c588be-8302-1222-4cba-
51c8625a9811","name":<script>alert(1)</script> ...}
```

The form was properly sanitized since the script did not execute. As can be seen in the output below, the page showed an error message and then aborted the change. It should be noted that it was not possible to get attacker-controlled content into the pop-up.



Id: 62c588be-8302-1222-4cba-51c8625a9811 required
A string uniquely identifying this node. If left empty a GUID will be generated for Id.
Parent: 92b92c88-16a5-4f72-a927-000405e27d1c Change
A URI specifying the parent element. If left empty it is a top-level element
URI: asclepios:62c588be-8302-1222 required
A URI uniquely identifying this node. If left empty ... efix.
Type: CONCEPT required
The type of this node. Allowed values: CONCE
Name: ContextPattern required
The display name of the node
Description: Context patterns are recurring motives of data access occurences that are recognized in repeating context elements instances.
An explanatory description of the node's purpose

```
STATUS=error
ERROR=
```

I.3.3. *Input sanitization*

The most common web application security weakness is the failure to properly validate input coming from the client or from the environment before using it. This weakness leads to many major vulnerabilities in web applications.

When testing the model editor, undefined behaviour was caused when a field was changed to the double quote character (").

If this following change was applied, then the editor was unable to load any attributes.

```
 POST /gui/admin/save-attribute HTTP/1.1
 Host: localhost:9090

 {"id":"1f74c52f-2e89-4b01-9b7d-0d405724d673","uri":"asclepios:1f74c52f-2e89-4b01-9b7d-
0d405724d673","type":"CONCEPT","name":"\"", [redacted]
```

It can be seen from the application output that the double quote character was not properly escaped and was causing this behaviour.

```
2020-03-23 11:22:28.557 DEBUG 5313 --- e.a.ample.gui.AttributesController       : -------------
getTopLevelAttributes: OUTPUT: [ { "id":"92b92c88-16a5-4f72-a927-000405e27d1c", "text":""",
"children":true, "icon":"/images/concept.png", "type":"CONCEPT" }]
```

There were attempts to exploit this behaviour by using strings that are normally used to identify possible XSS and SPARQL injection locations, without success.

Finding 1
/forms/admin/schema-mgnt.jsp

Finding:
Input data is not properly sanitized.

Confirmation & Substantiation:

Changing a field to a double quote character (“) is accepted and causes the application to have undefined behaviour. After applying this change, the editor is unable to load the attribute that contains the changed field and also all subsequent attributes.

Recommendation:
Validate all input which originates from the user, including fields which are not normally visible in the user interface.

I.3.4. HTTP Verb Tampering

The HTTP specification includes request methods other than the standard GET and POST requests. A standard compliant web server may respond to these alternative methods in ways not anticipated by developers. Although the common description is ‘verb’ tampering, the HTTP 1.1 standard refers to these request types as different HTTP ‘methods.’
In order to test, we modified requests sent by the editor and examined its behaviour. Header methods OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT, FOO were tested for each endpoint.

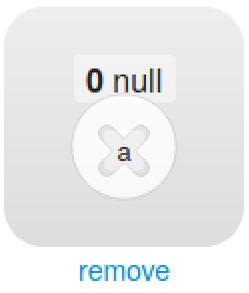
By default, endpoints were called with a GET request. Methods ‘OPTIONS’ and ‘HEAD’ were accepted server-wide. The endpoints that concerned the Model editor form and import/export functions required a POST request.

Requests made using the known methods (DELETE, PUT, CONNECT, TRACE) were correctly rejected and returned HTTP/1.1 405 Not Allowed. Using the non-existing ‘FOO’ method, the server returned HTTP 500 Internal Server error. This behaviour is intended.

I.3.5. File upload

The editor offers the functionality of uploading a user provided file to be imported to the editor. When applications offer file uploads, it is important to enforce restrictions to the file extension and also the content. For instance, if HTML or TXT files are allowed, XSS payload can be injected in the file uploaded.
In the case of the model editor, it was observed that only TTL extension files were permitted. After some testing, it was determined that the content of the files is not being checked before being uploaded to the server and applied to the editor.

First, the behaviour of the editor was examined to see if there was a check on the file extension that was being uploaded. The server rejects the file as can be seen in the output below.

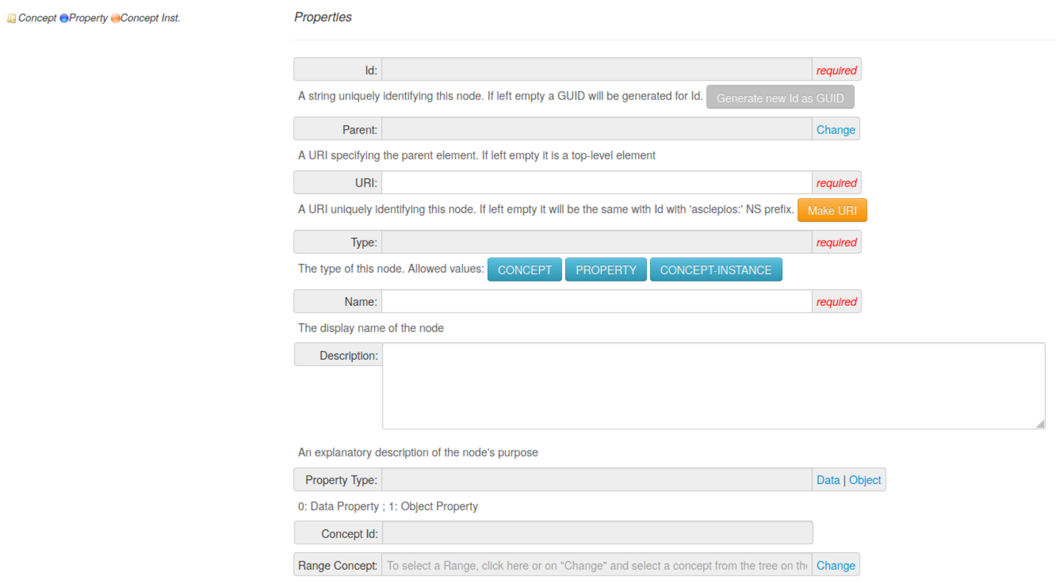


The rest of the tests concerned files with the TTL file extension. A TTL file could be uploaded to the server with the following content:

```
<http://www.asclepios.eu/metadata-schema/ASCLEPIOS-OBJECT#c1e7ea6d-c892-42f1-a146-8c7497fd4bb6>
      a                         <http://www.asclepios.eu/metadata-schema#ASCLEPIOS-OBJECT> ;
      <http://purl.org/dc/elements/1.1/type>
```

```
        "CONCEPT" ;
    <http://purl.org/dc/terms/URI>  "asclepios:c1e7ea6d-c892-42f1-a146-8c7497fd4bb6" ;
    <http://purl.org/dc/terms/created>
        "1900-01-01T00:00:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
    <http://purl.org/dc/terms/creator>
        "" ;
    <http://purl.org/dc/terms/description>
        "Velocity" ;
    <http://purl.org/dc/terms/identifier>
        "c1e7ea6d-c892-42f1-a146-8c7497fd4bb6" ;
    <http://purl.org/dc/terms/modified>
        "1900-01-01T00:00:00Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
```

As the closing dot is missing, this file is not correct, and the editor has issues processing this file.



Concept Property Concept Inst.

Properties

Id:		required
A string uniquely identifying this node. If left empty a GUID will be generated for Id.		Generate new Id as GUID
Parent:		Change
A URI specifying the parent element. If left empty it is a top-level element		
URI:		required
A URI uniquely identifying this node. If left empty it will be the same with Id with 'asclepios:' NS prefix.		Make URI
Type:		required
The type of this node. Allowed values:	CONCEPT PROPERTY CONCEPT-INSTANCE	
Name:		required
The display name of the node		
Description:		
An explanatory description of the node's purpose		
Property Type:		Data | Object
0: Data Property ; 1: Object Property		
Concept Id:		
Range Concept:	To select a Range, click here or on "Change" and select a concept from the tree on th	Change

It can be seen that the editor can no longer load the attribute tree on the left and is unusable.

Finding 2
/forms/admin/import.jsp

Finding:
User provided file is not checked before upload and is added to the editor.

Confirmation & Substantiation:
Even though there is a check for the file extension to be 'TTL', the contents of the file are not checked. If the structure of the TTL file is not as expected, it is still uploaded to the editor. When the editor tries to display the contents, this causes it to have undefined behaviour.
See Listing 2 in page 3 which shows the state of the editor after a corrupt file is uploaded.
Listing 3 shows an example of a file that can be uploaded which can cause the editor to have undefined behaviour. Note that only a final '.' (dot) is missing from the file.
Recommendation:
Validate the file extension and content of user provided files.

Note: From a technical point of view we consider this to be a relevant finding for the functionality and security of the editor. At the same time, after confirming with the Institute of Communication & Computer Systems (ICCS) - the developers of the editor – it is assumed that the operators of the editor are trustworthy individuals, therefore mitigating this risk. Due to this assumption, the impact of this finding can be skipped in the overall security of the editor. For traceability purposes, this finding is kept in the report.

I.3.6. File Export

The editor offers the download of the complete ruleset / model as one single file with the TTL extension. Usually when such functionality is offered, the request to download can be manipulated into retrieving files different than the ones intended by the application.
In the case of the editor, as there are no parameters being passed to the server, no manipulation is possible to include local system files or other sources.
Therefore, we do not identify any security vulnerabilities in this function.

I.4. Summary of Findings

Finding 1
/forms/admin/schema-mgnt.jsp

Finding:
Input data is not properly sanitized.

Confirmation & Substantiation:
Changing a field to a double quote character (“) is accepted and causes the application to have undefined behaviour. After applying this change, the editor is unable to load the attribute that contains the changed field and also all subsequent attributes.

Recommendation:
Validate all input which originates from the user, including fields which are not normally visible in the user interface.

Finding 2
/forms/admin/import.jsp

Finding:
User provided file is not checked before upload and is added to the editor.

Confirmation & Substantiation:
Even though there is a check for the file extension to be ‘TTL’, the contents of the file are not checked. If the structure of the TTL file is not as expected, it is still uploaded to the editor. When the editor tries to display the contents, this causes it to have undefined behaviour.
See Listing 2 in page 3 which shows the state of the editor after a corrupt file is uploaded. Listing 3 shows an example of a file that can be uploaded which can cause the editor to have undefined behaviour. Note that only a final ‘.’ (dot) is missing from the file.
Recommendation:
Validate file extension and content of user provided files.

Note: With respect to Finding 2, please refer to section I.3.5 for details on why this finding is mitigated by the assumption of trusted application administrators.