



EESSI - RINA

Identity and Access Management (IAM)

Architecture & Design Specifications

Table of Contents

Table of Contents.....	2
1 EESSI Identity and Access Management Overview	6
2 Functions and use cases	8
2.1 IAM configuration.....	8
2.2 IAM authentication.....	9
2.3 External - Internal repository synchronization	10
2.4 External rules for process assignment	11
2.4.1 Execution of external rules.....	11
2.4.2 Logical components	12
2.4.3 Illustrative scenario example – external filtering rules.....	14
2.4.4 XACML Concepts	15
3 Behaviour	17
3.1 IAM authentication.....	17
3.1.1 Internal User Authentication.....	19
3.1.2 External User Authentication – LDAP reference implementation	19
3.2 LDAP- synchronization.....	21
4 IAM external interfaces.....	22
5 Configuration of SAML 2.0.....	23
5.1 Limitations.....	23
5.1.1 IdP SingleSignOnService Binding.....	23
5.1.2 User Identification in RINA Context.....	23
5.2 Workflow	23
5.3 Configuration.....	25
5.3.1 CAS Configuration	25
5.3.2 RINA Configuration	26
6 Configuration for JWT Authentication	27
7 Configuration of Single-Sign-On.....	29
8 Other Authentication Plugins.....	30
8.1 Other Authentication Plugins.....	30
9 Samples	32
9.1 Update LDAP connection settings request.....	32
9.2 Get LDAP connection settings response.....	32
9.3 Update LDAP group parameters mapping request.....	32
9.4 Get LDAP group parameters mapping response	32
9.5 Get LDAP group parameters keys.....	33
9.6 Update LDAP user parameters keys request.....	33
9.7 Get LDAP user parameters keys.....	33
9.8 Get LDAP user parameters keys.....	34
9.9 Synchronize LDAP.....	34
9.10 Reassign group affected cases.....	34
9.11 Retrieves a list of policies, optionally filtered	34
9.12 Retrieves a policy.....	35
9.13 Creates a policy.....	36
9.14 Updates a policy.....	36
9.15 Deletes a policy	37
9.16 Associates an existing policy to a target.....	37
9.17 Associates existing policies to a target.....	37
9.18 Removes a target from a policy	37
10 PDP - Case Assignment Policy Decision Point.....	38
10.1 Service contracts / interface	38
10.2 Data contracts / model.....	38
10.2.1 Actor.....	38
10.2.2 UserOrGroup.....	38

10.2.3	Case properties.....	38
10.2.4	Participant.....	39
10.2.5	Organisation (inherits Entity).....	39
10.2.6	Person (inherits Entity).....	39
10.2.7	Entity.....	39
10.2.8	Address.....	39
10.3	Fault contracts / exceptions.....	40
11	Development impact.....	41
12	How-to's.....	42
12.1	How to configure the synchronizer.....	42
12.2	How to develop my external rules	42
12.3	How to configure LDAP.....	44
12.3.1	Authentication.....	44
13	Relevant links & References.....	45
14	Limitations of the current version.....	46

Document Control Information

Document Control	Value
Project Title	Electronic Exchange of Social Security Information (EESSI)
Document Name	EESSI – RINA - Identity and Access Management (IAM)
Document Category	Architecture & Design Specifications
Revision	-
Component Version	-
Last Publication Date	22/12/2020
Project Milestone	EESSI-2020
Document Status	Final
Sensitivity (TLP) Distribution terms	Traffic Light Protocol (TLP) = "GREEN" The distribution of this document is done strictly in line with the Traffic Light Protocol (TLP) established by the European Commission's note AC 790/15 REV for the EESSI project documentation. In line with the note AC 790/15 REV, this document is labelled as TLP = "Green". Therefore, it can be circulated widely within the EESSI community. However, the document or the information herein may not be published or posted on the Internet, nor released outside of the EESSI community.
Connected/Embedded Files	None
Authors	European Commission, DG EMPL F5, EESSI ARCH/RINA
Revised by	European Commission, DG EMPL F5, EESSI QA
Approved by	European Commission, DG EMPL F5, EESSI PM

Document history

Released Component version	Changes/Corrections
	Description
EESSI-2019 / RINA 5.6.2	Added a new method in the service contract
	Remove chapter 4
	Add chapter 7 for SSO configuration. The detailed technical information about Data and Services contracts are presented in the CPI reference guide
EESSI-2020 / RINA 6.2.1	Description updated to reflect then new RINA developments and architectural improvements

1 EESSI Identity and Access Management Overview

Part of the National Institutions Domain, RINA (Reference Implementation for National Applications) consists of a collection of infrastructure and communication services, foundation, repository and publishing services, business, integration and user interface services which will provide for clerks and their organizations, the tools to implement the EESSI specific International Protocol of Data Exchange based on Structured Electronic Documents in Social Security belonging to European Community Member States and Associated States.

RINA is based on 4 layers which are built one on top of the other, plus the CAS module:

- *Technical Message Services (TMS)*. - provide the mechanism for sending/receiving technical messages to the Access Point. It works based on the protocol ebMS3.0 - AS4 profile. It is an internal layer and cannot be used by third parties, being its unique client the RINA BMS.
- *Business Messaging Services (BMS)* - provide a reusable component for sending business messages to the Access Point via the TMS layer. These services provide the translation (business message to technical message and vice versa), transformation (converting IDs to GUIDs), validation and signing of the business messages for correct exchange with the EESSI environment, and the correct reception of messages from other institutions. It offers the BMI WS interface making available the integration of third party applications to the EESSI ecosystem.
- *Case Processing Services (CPS)* - provide a state full component built on top of the Business Messaging Services that manage cases in a structured manner taking care of all the issues regarding case flow, documents, notifications, user management and provides two interfaces for external access (NIE and CPI)
- *Portal* - provides an UI build on top of the Case Processing Services (CPS) having administration consoles and case Processing modules
- *CAS Authentication Server* - the SSO (Single Sign On) server providing the authentication to users/clients requesting to use a service

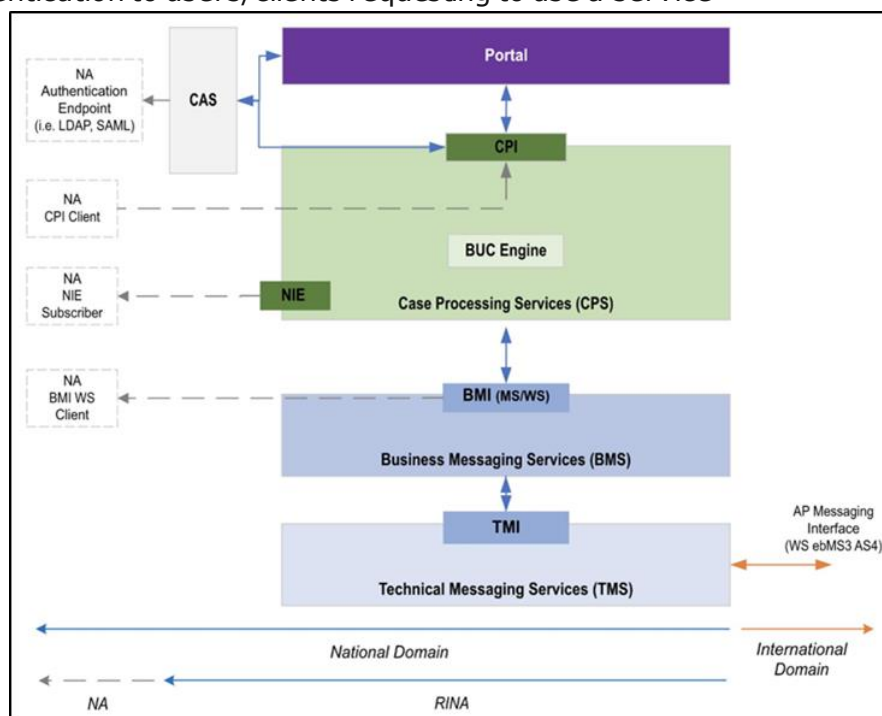


Figure 1. Rina Functional Layers, modules and Interfaces

The scope of this document is to provide a specification document for the Identity and Access Management Interface, part of the Case Processing Services (CPS).

Here is a short description of how this document is structured :

- Interface overview – this chapter, an introduction to what this interface is about, where its place is and why it is needed;
- Functions and use cases – a presentation of high level functions and use cases that drive this service component;
- Behaviour - Sequence diagrams – behavioural diagrams that describe the flow between components;
- IAM - Service contract – a list of REST operations that make up the API;
- IAM - Data contract – the REST data model used for this service;
- Samples – some samples that show how to use this API;
- Development Impact;
- How to's;
- Relevant links.

This interface provides a RESTful web service that allows the RINA portal or national systems to interact with the Identity and Access Management service that is one component part of the Case Processing Services.

The Identity and Access Management service provides the following capabilities:

- Central Authentication Server (CAS) – provides a Single-Sign-On server for authenticating to RINA services (currently only CPI). This provides also a functionality for SAML 2.0 delegated authentication, LDAP authentication, etc;
- Identity provider configuration – it is possible to choose between an internal and an external identity provider implementation. RINA offers an LDAP integration as a reference external identity provider implementation;
- Authentication – the users are authenticated against the identity provider configured in CAS;
- Identity synchronization interface – an interface for synchronization the users from the external source to the internal environment;
- External rules for process assignment – an API for setting up rules that say who can manage what case type depending on the rules that can be configured externally.

2 Functions and use cases

The Identity and Access Management top-level function is a collection of services that enables the system to securely control access to RINA resources for internal or external users.

It comprises four basic functions as shows in the diagram:

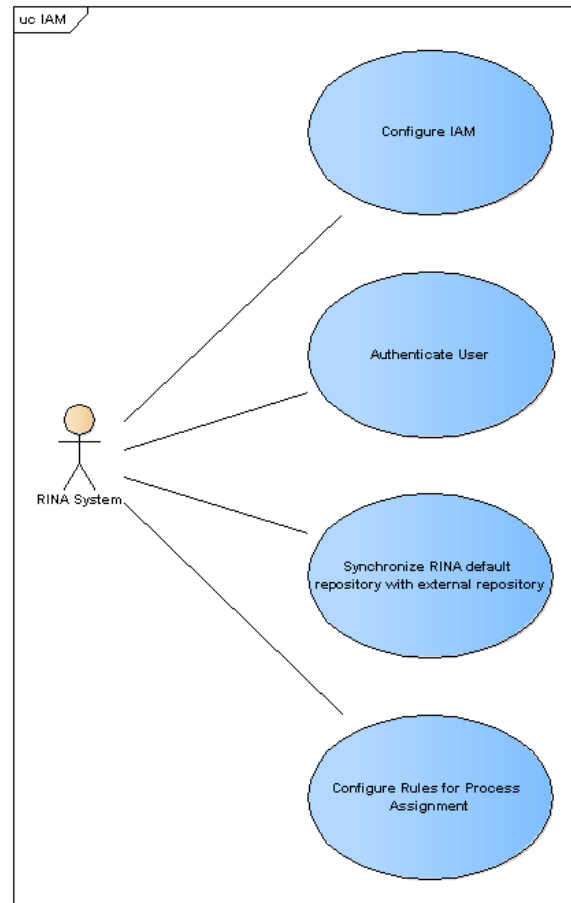


Figure 2. Identity and access management use cases

A short description for each basic function:

- IAM Configuration - The RINA IAM can be configured to use the internal identity provider that is the system default source or an external data source;
- User authentication - When the user logs into the system, he must be authenticated against the configured identity provider;
- External-Internal synchronization – When external identity provider is selected for managing the organization, a tool synchronizes all the users and groups from the external repository with the internal repository;
- External rules for process assignment – Rules that say who can manage what process;

2.1 IAM configuration

The RINA system administrator must be able to configure the Identity and Access Management module.

This module has two major configuration parts as depicted in the use case below.

The 'Configure Authentication' use case enables the RINA system administrator to choose the identity provider from one of the options available. RINA currently provides following authentication providers:

- Internal (default) – the users are authenticated against their username/password stored in RINA datastore.
- LDAP provider – the users are authenticated against an external LDAP server. The user/group profiles are still to be stored (synchronized) in the internal datastore, however the passwords are not needed.
- SAML 2.0 provider – enables to authenticate the users against an external SAML 2.0 identity provider. The user/group profiles are still to be stored (synchronized) in the internal datastore, however the passwords are not needed.
- JWT Token – enables the authentication through a self-containing Java Web Token.

It is important to note that all external identity providers are used for the user authentication only. The user and group profiles must still be stored in the RINA datastore. RINA provides a synchronization functionality to be able to create/update these profiles from external databases.

There are no restrictions on the supported external identity providers, however, they must be accessible from the RINA infrastructure (i.e. firewalls settings, etc.).

The 'Configure Synchronizer Tool' use case sets up the configuration for a tool that synchronizes users and groups profiles from the external LDAP repository into the default repository. Synchronization from other repositories (like SQL) are not provided out-of-the-box but there is an import API which can be used. Another possibility is also to synchronize the users/groups profiles directly into the Rina datastore.

The following properties should be at least specified for the user synchronization tool:

- Server URL – the URL of the external repository server;
- User credentials – username and password for accessing the external identity provider;
- LDAP user search base – organization and domain in LDAP server for users synchronization;
- LDAP group search base – organization and domain in LDAP server for groups synchronization;
- Security authentication – type of security used in given LDAP server;
- User and group filters – provide a way to filter all the users and groups.

Apart from this, mappings between internal user and group properties and LDAP attributes should be provided.

2.2 IAM authentication

This use case is a prerequisite of performing any actions in the RINA application.

The user authentication ('Authenticate User' use case) is done either against the internal repository or against an external identity provider as depicted below:

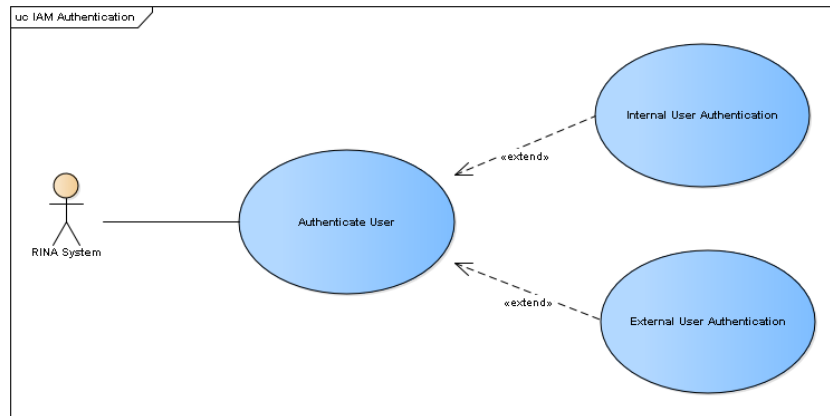


Figure 3. IAM Authentication use case

The 'Authenticate User Internal' extended use case implements the use case where the user logs into the system as a known RINA user and both the username and password can be verified in the local repository.

The 'Authenticate User External' extended use case implements the use case where user logs into the system as a known external repository user and only the username can be verified in the local repository. The internal repository does not store any password or other cryptographic material, so the credentials validation takes place against an external data source via a connector.

The reference external authentication implementation offered by RINA is currently one of the following:

- LDAP
- Delegated SAML 2.0 Identity Provider
- JSON Web Token

2.3 External - Internal repository synchronization

In the case of the external authentication, the RINA system must synchronize the users and groups from the external source as shown below:

The synchronization takes place in one direction, from the external source to the internal destination, so the external repository is never changed.

The synchronization tool must perform all the CRUD (Create, Read, Update, Delete) operations in the internal repository so that the users and groups match as close as possible the attributes from the external source.

The synchronization tool connects to both sources and performs the following actions:

- It reads all the users from external repository and creates them in the internal repository;
- It reads all the groups from the external repository and creates them in the internal repository;
- Updates all the memberships (user, group, role) in internal repository to match the external membership.

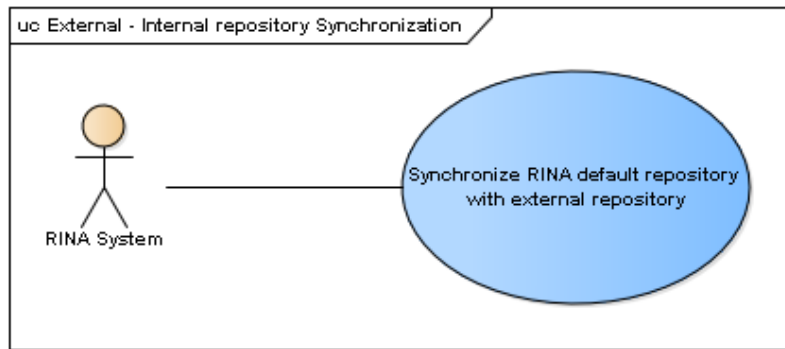


Figure 4. External - Internal repository synchronisation

2.4 External rules for process assignment

There are two parts to consider when dealing with the process assignments:

- Configuration – the management part where the rules for assigning are created and setup;
- Execution – the action part, when the filter rules are being applied on the process.

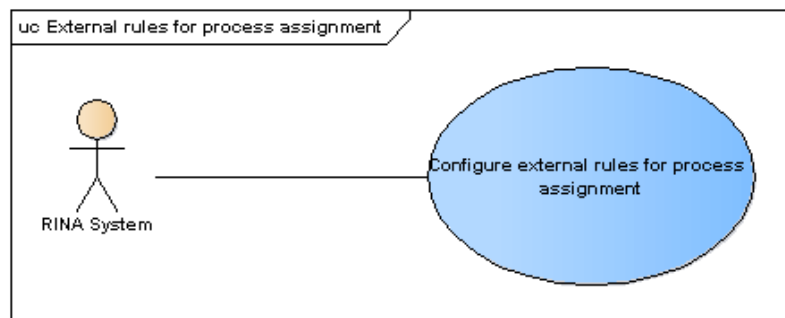


Figure 5. External rules for process assignment use case

2.4.1 Execution of external rules

The execution of external rules is triggered as shown in the figure below, when a case is received or created.

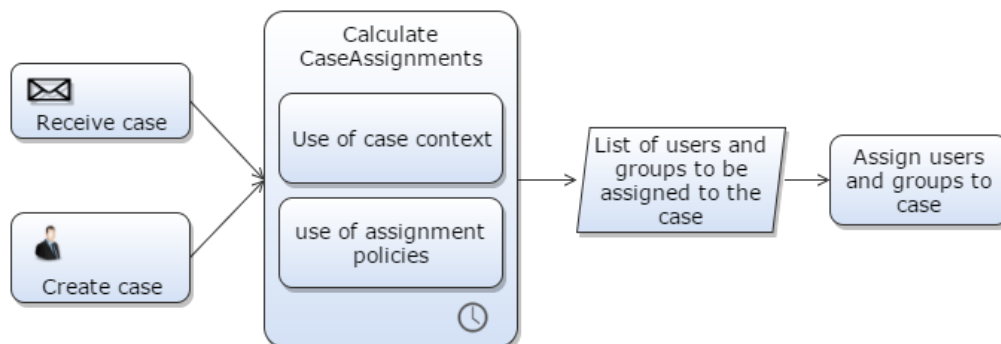


Figure 6. Execution of external case assignment rules

This triggers a method which calculates the case assignments based on the case context and the defined assignment policies.

Notes:

- The assignment policies are defined by RINA administrator initially before users start to use RINA;

- The assignment policies / automatic rules are defined by RINA administrator;
- The user assignments created automatically can be later edited / changed by supervisors of each case;
- The case assignment policies are used to assign users and/or groups to individual cases when the cases are created (either locally, either received from another party); but to decide which users are allowed to create different types of cases in the first place, a special type of policies are used: "Creator policies". These have the same structure and are managed the same way as the regular policies;
- An administrator can decide to create as many policies as needed, but only policies that are actually assigned to targets are taken into account when calculating the assignments.

More technical details can be consulted in section [10 PDP - Case Assignment Policy Decision Point](#).

2.4.2 Logical components

The process assignments have the following logical components:

- Sectors – these are predefined sector areas the processes are being part of. (For instance Pension, Sickness, Family Benefits are sector examples);
- Process – each sector is driven by one or more business processes (for example in the Pension sector there are currently defined several case types, "Invalidity Pension Claim" and "Old Age Pension Claim" being two of them);
- Application roles – Each process has two application roles:
 - Process owner – this is the party that creates and starts the whole case;
 - Counterparty – this is the party that receives the initial case;
- User/ actor - An actor is a placeholder specified in the process definition for the users. Actors are the individuals called to complete a Task (they become – potentially different - users in the course of each case). All human tasks have at least one actor assigned;
- Group – A set of individual users, defined by the Administrator;
- Membership – Users which are belonging, either individually or collectively, to a group with a specific role within it;
- Sender country;
- Assigned users/groups – the users or group of users that can be assigned for each type of actor;
- Role – The title/position/job of an individual user. The user roles consists in the following:
 - Manager/Supervisor;
 - Authorised Clerk;
 - Un-authorised Clerk;
 - Auditor;
 - Viewer;
 - Medical User;
 - VIP User;

The actions that the above described above user roles can perform are described in the table below:

RINA USER ROLES		Manager/ Supervisor	Authorised Clerk	Un- Authorised Clerk	Auditor	Viewer	Medical User	VIP User	Everyone
Case Assignment	Assign Case	✓							
	Request Assignment	✓	✓	✓	✓	✓	✓	✓	✓
	Authorise Assignment	✓							
Case Management	Create Case		✓	✓					
	Create SED		✓	✓					
	View SED	✓	✓	✓	✓	✓			
	Send SED	✓	✓						
	Request Approval for sending			✓					
	Approve and Send SED	✓	✓						
	Upload/ Delete/ Send/Download medical attachments						✓		
	View VIPS							✓	
	Change Case Metadata*	✓	✓	✓					
	Manual case archiving	✓	✓						
	Manual case unarchiving	✓	✓						
	Set/clear alarms	✓	✓	✓					
	View case metadata* ¹	✓	✓	✓	✓	✓	✓	✓	✓
	Download/send regular attachments	✓	✓	✓	✓	✓			
	Add-Upload/delete regular attachments		✓	✓					
	Add/delete comments	✓	✓	✓	✓	✓			
Audit	View Audit	✓			✓				

For automatic process assignment, external rules filtering policies can be used.

* i.e. Subject block (username, name, gender and birth date or the relevant institution description in bilateral reimbursement cases), Case participants & assignments, document list (SEDs), alarms, SED and Case level comments

Notes:



- *A User/Clerk can have multiple roles. A user with multiple roles can perform any action permitted by all the assigned roles. As an example, an authorised clerk that is also a medical user can send SEDs (permitted by the authorised clerk role) and also read classified medical attachments (permitted by the medical user role);*
- *In general, Medical and VIP roles have limited permissions (view metadata and handle Medical and VIP data) and can be considered as supplementary roles that are assigned to specific users when access to critical data is required;*
- *Only VIP users are able to select/unselect the "Sensitive" flag in the "Case Metadata" popup;*
- *VIP user may set "Sensitive Case" flag BUT in order to create/edit documents must be also authorised/non authorised clerk. In order to send documents the clerk must be authorised. In order to view case documents, the clerk must be authorised/non authorised, supervisor and/or viewer*
- *Medical user can tag/untag an attachment as medical. In order to add must be also authorised/non authorised clerk. In order to view it, the user must be authorised/non authorised, supervisor and/or Viewer*
- *The roles Supervisors, Authorised clerks and non-authorised are able to change the Criticality and Importance properties (Case Assignments)*
- *Some groups can be labelled with the "Organisational Unit" flag (the equivalent of the OU in LDAP). This is used in order to represent the branches.*

2.4.3 Illustrative scenario example – external filtering rules

For external filtering rules, an illustrative example for each application role is given below:

Process owner example

The organization has a number of branches.

Each branch has its own departments in the areas of the well-known sectors.

Each department has people that can be part of other branches as well.

One external rule that must be supported is the following:

- When a user creates a case all the users from the group the owner is part of should be assigned automatically for that case. If the user creator is part of more groups (for example the user is both in the Pension and Sickness sectors) the group should be selected.

Counterparty example

The organization has a number of user experts that can deal with cases from specific parts of the world.

One external rule that must be supported is the following:

- When a case arrives from Germany, all the users from the Western Europe group are automatically assigned to the case.

2.4.4 XACML Concepts

The concepts used in the current document for the external rules for process assignment consist in a re-interpretation of the XACML concepts.

A mapping to the XACML concepts is introduced in the table below.

XACML concept	RINA concept
PAP Policy Administration Point.	Assignment Policies Administration: the /AssignmentPolicies REST API & the Process Assignments tab from RINA Admin Portal.
Policy A Policy represents a single access control policy, expressed through a set of Rules.	Policy
PolicySet A PolicySet is a container that can hold other Policies or PolicySets.	Policy group.
Rule A policy can have any number of Rules which contain the core logic of an XACML policy.	Rule.
Target Basically a set of simplified conditions for the subject, resource, and action that must be met for a policy set, policy, or rule to apply.	Target: A fragment of the process definitions tree to which a policy can be associated to. (i.e. Sector, Process Definition, Application role, etc.)
Subject The entity requesting access.	The collection of groups/users that will be assigned to a specific actor.
Resource The resource that is being accessed.	The case instance.
Action The type of access requested on the resource.	The assignment as an actor to the case instance.
Condition A Boolean function. If the Condition evaluates to true, then the Rule's Effect is returned.	Condition: a collection of 0 or more conditions, which must be met in order for a rule to be applied.
Combining algorithms This algorithm is responsible for combining outcome from multiple policy evaluations.	There are only use 3 types of combining algorithms: - the rules inside a policy are always combined with the "first matching rule" algorithm;

	- the policies, except "<i>Creator Policy</i>" inside a policy group are always combined with the "intersect outcomes" algorithm; The intersection combination is applicable inside the same group; - when a target has more policies or policy groups assigned to it , there is an implied combining algorithm of the results: "union outcomes";
Effect Permit/Deny.	In RINA, all rules have an implied "permit" effect.
PDP Policy Decision Point.	Case Assignment Policy Decision Point It evaluates the assignment policies for each case instance and decides which users/groups will be assigned as actors of this case instance. (see section 10 PDP - Case Assignment Policy Decision Point)

3 Behaviour

RINA is shipped with a Central Authentication Server (CAS), which handles the authentication of a user. Internal, as well as external authentication is always managed through the CAS server. Below, you will find the authentication schema used by RINA CPI (and possibly other system components).

The CAS server acts as a single-sign-on server, similar to the former OAuth 2 authentication server, however it provides many out-of-the-box integrations with third party systems and authentication protocols.

3.1 IAM authentication

This diagram shows the general flow for user authentication, from the moment the client initiates the login until it receives the authenticated session cookie. It displays the sequence of the steps, independent from the type of authentication that is used.

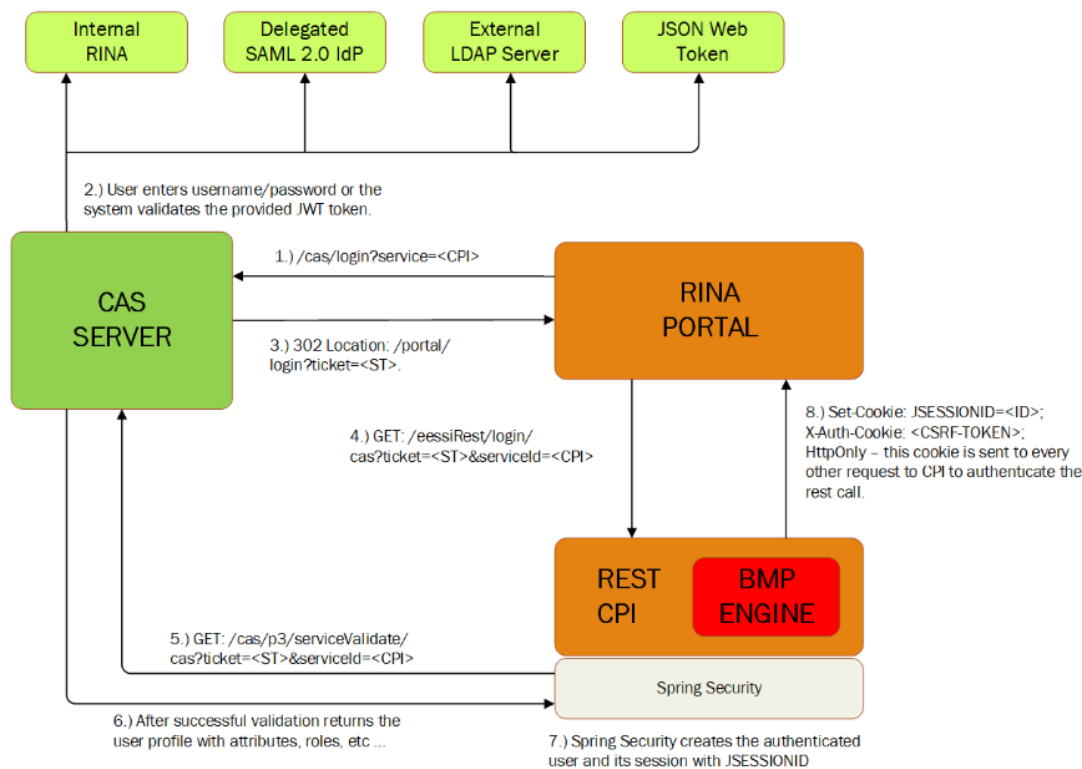


Figure 7. IAM Authentication sequence diagram

The RINA CPI uses the CAS 3.0 protocol for authentication (specification can be found here: <https://apereo.github.io/cas/6.2.x/protocol/CAS-Protocol.html>). Following parties are present in the scenario:

- CAS Authentication Server – the SSO CAS server providing the authentication of users which want to use a given service.
- Service – CAS is providing the authentication for registered services. In our case, the registered services are:
 - RINA REST CPI module
- Portal/UI – is the client, which tries to access the service and needs to obtain an authentication from the CAS server.

The communication with the BPM Engine is executed through a single system user.

To understand the interaction of the parties, it is first required to define a couple of terms from the CAS 3.0 protocol. The protocol can be viewed similar to OAuth 2.0, however, instead of tokens the focus is on tickets. CAS issues 2 types of tickets for our context:

- Ticket-Granting-Ticket (TGT) – is generated by the CAS server after a successful authentication and this ticket is a long-term ticket. TGT is used for generating service tickets (ST), which are used for authentication in the service. The service itself is agnostic of any credentials of the authenticated user.
- Service Ticket (ST) – is generated by the CAS server for the client, which has already been authenticated (i.e. has obtained a TGT) when trying to access the service. ST is a short-term valid ticket (default is 10 seconds) and can be validated only once. Any ST, which has already been validated would fail in another validation.

After successfully obtaining a service ticket, this must be validated by the service and if it is valid, an authenticated and authorized session is created associated with a session ID.

Following is the description of the authentication process as shown on the figure 7:

- **Step 1:** RINA Portal (or any other client) wants to access the RINA CPI (the service registered in CAS). It redirects the user to the CAS server with the service ID parameter.
- **Step 2:** The user authenticates against the configured identity provider (RINA internal, LDAP, SAML 2.0 or by a JWT token). If the user is authenticated successfully, a SSO session in the CAS server is created (represented by the TGT).
- **Step 3:** The CAS server generates a service ticket (ST) for the service ID sent in the request in the point 1. The service ID is also the URL which receives the ST in the HTTP 302 Redirect response. The user is redirected to the provided service ID URL with the generated service ticket.
- **Step 4:** The client calls a "login" method of the RINA CPI with the service ticket and the service ID, which was used to obtain a service ticket.
- **Step 5:** The RINA CPI security module authenticates the received service ticket by calling the CAS server API method. To do this, it sends the ST together with the retrieved Service ID to the validation endpoint of the CAS server.
- **Step 6:** The CAS Server returns the authentication response, which contains the user ID (optionally also other attributes).
- **Step 7:** The security module in the RINA CPI tries to find the profile for the received user ID and creates an authenticated session.
- **Step 8:** The security module in the RINA CPI returns back to the client an HTTP-only cookie with the name JSESSIONID, which must be sent with any subsequent request to the RINA CPI.

The CAS server is provided as **eessi-cas-server** maven project and is built as a deployable WAR file. It can be deployed in the same server as the RINA CPI module or on a separate one. The maven project contains the custom internal authentication module and accesses the RINA datastore through a dependency on the RINA services.

Important to notice: the CAS protocol uses HTTP redirects or other HTTP communications, which can contain the TGT or ST. None of those should ever get disclosed. From this reason, it is always **recommended to use SSL/TLS for securing the communication channel**.

3.1.1 Internal User Authentication

As stated earlier in this document, the internal authentication (as well as the external) is always managed by the CAS server. The difference lies in the storage of the authentication material. The internal user authentication authenticates users against their profiles stored in the RINA internal database.

To successfully authenticate a user, the authentication module needs the username and the password (RINA only stores a hash of a password and a random salt used with the hashing function). Once the user enters the username and password following steps are performed by the **EessiUserAuthenticationHandler**:

- Find a user with by the provided username, which is not marked as deleted. If the user is not found an exception is thrown.
- The provided password is hashed together with the salt stored in the user document. This hash is then compared with the one stored in the user document. If it does not match an exception is thrown.
- The "isEnabled" flag on the user document is checked. If the value is false an exception is thrown.
- The "isLocked" flag on the user document is checked. If the value is false an exception is thrown.

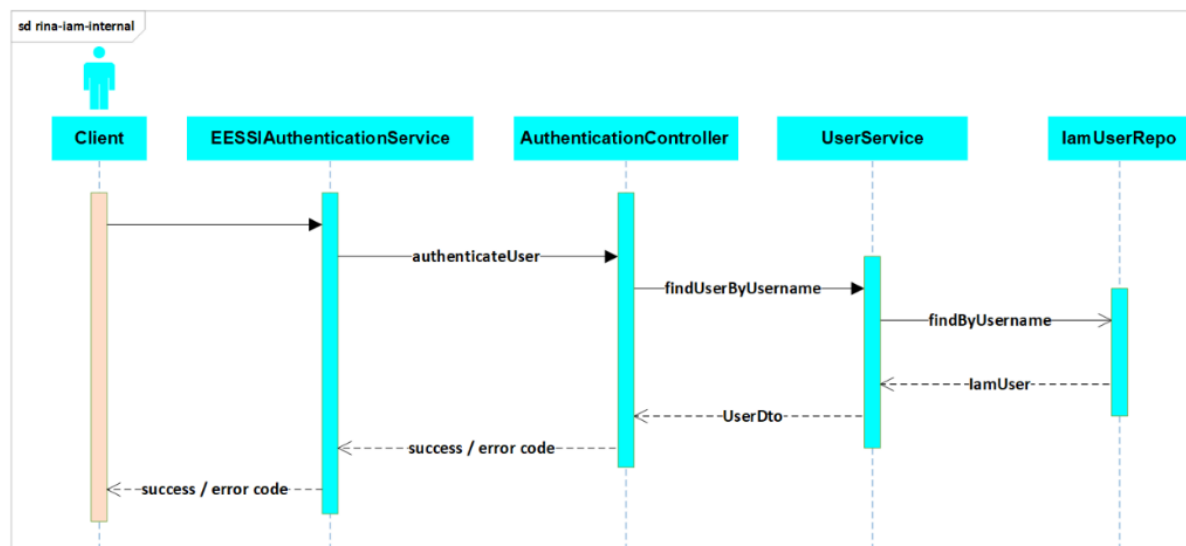


Figure 8 depicts the sequence diagram when using the internal authentication in the context of the RINA services.

Figure 8. Internal user authentication

3.1.2 External User Authentication – LDAP reference implementation

This sequence diagram is a zoom-in for the external IAM authentication with LDAP diagram that shows how the EESSIAuthenticationService is implemented to give access to an external data source.

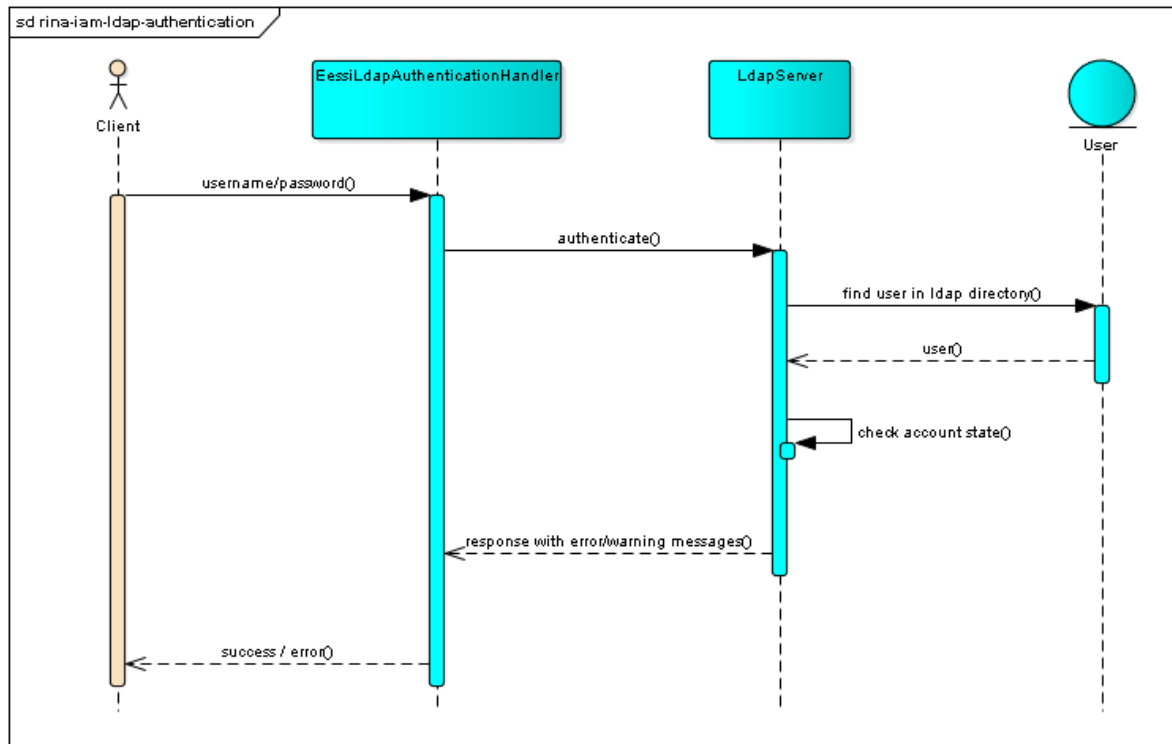


Figure 9. External user authentication

- Based on configuration of authentication handler in cas.properties file it is determined what kind of authentication is used.
- It is possible to use internal (PostgreSQL) and external (LDAP) authentication in the same time.
- It is also possible to use multiple LDAP sources. In this case authentication service iterates over defined sources and uses the first one which successfully authenticates user.

If authentication is not successful in any of the sources, exception is thrown.

3.2 LDAP- synchronization

This sequence diagram shows how the synchronization tool interacts with the LDAP source and PostgreSQL DB.

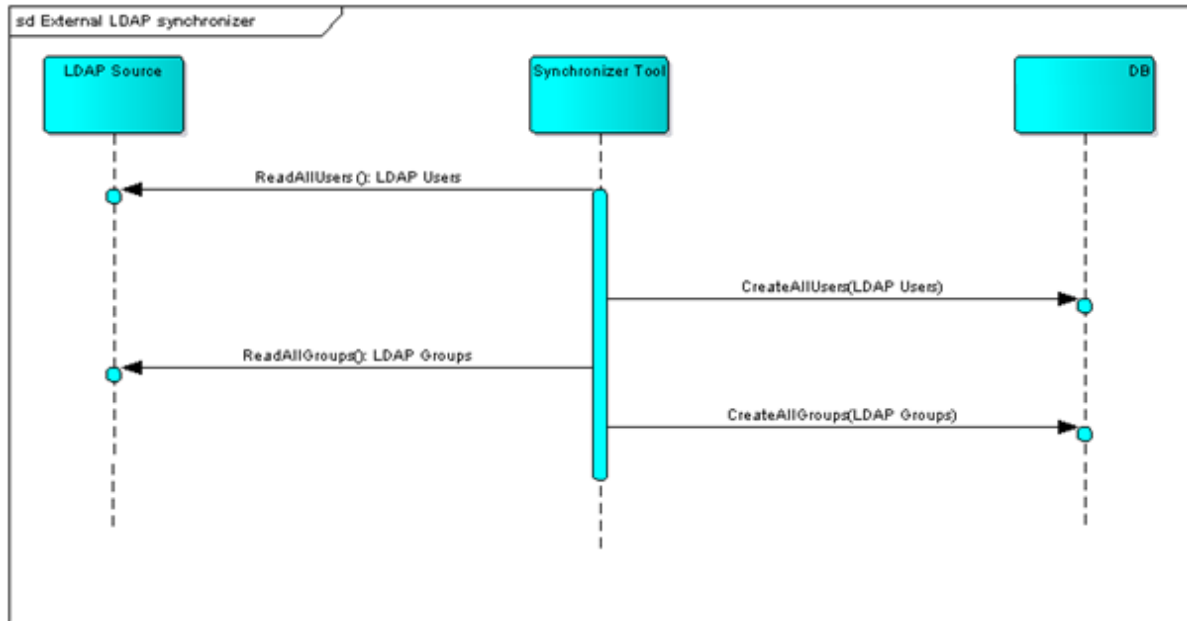


Figure 10. External LDAP synchronization

The synchronization takes place in one direction, from the LDAP source to the DB destination, so the LDAP directory is never changed. The steps from the diagram are explained below:

- Reads all Users in the source LDAP directory
- Creates or updates Users in PostgreSQL which origin is given LDAP configuration
- Reads all Groups in the source LDAP directory
- Creates or updates Groups in PostgreSQL which origin is given LDAP configuration
- Retrieves all Users that are belonging to the groups in the source LDAP directory
- Retrieves all Users that are belonging to the groups in PostgreSQL

Please note that if another external identity provider besides LDAP is used, this sequence diagram is still valid, just that instead of the LDAP source there will be the other external source.

NOTE:

The following special characters should be escaped when used in a distinguished name: Plus sign (+), Semicolon (;), Comma (,), Backward slash (\), Double quote ("), Less than (<), Greater than (>), Pound sign (#)

4 IAM external interfaces

RINA IAM supports the following external authentication providers:

- LDAP
- SAML 2.0
- JSON Web Token (JWT)

RINA offers a REST endpoint implementation as a part of the Case Processing Interface (CPI) for configuring the institution LDAP data source synchronization.

The actual authentication REST flow from the sequence diagram Figure 3 is described in the specifications document ***EESSI-RINA Case Processing Interface*** in the chapter **2.1 User Authentication**.

A detailed description of the Service and Data contracts is given in the automatically generated Technical reference guide ("*EESSI – RINA Case Processing (CPI) – Reference*") that provides all the necessary technical information about the REST API.

5 Configuration of SAML 2.0

The CAS authentication server provides an option to use delegated authentication in terms of the SAML 2.0 protocol. In this case a user can get authenticated by an external delegated SAML 2.0 identity provider.

5.1 Limitations

5.1.1 IdP SingleSignOnService Binding

RINA authentication server (CAS) currently supports delegated SAML2.0 IdP integration in HTTP-Redirect Binding, i.e. the identity provider meta-data must define at least one `SingleSignOnService` with following attributes:

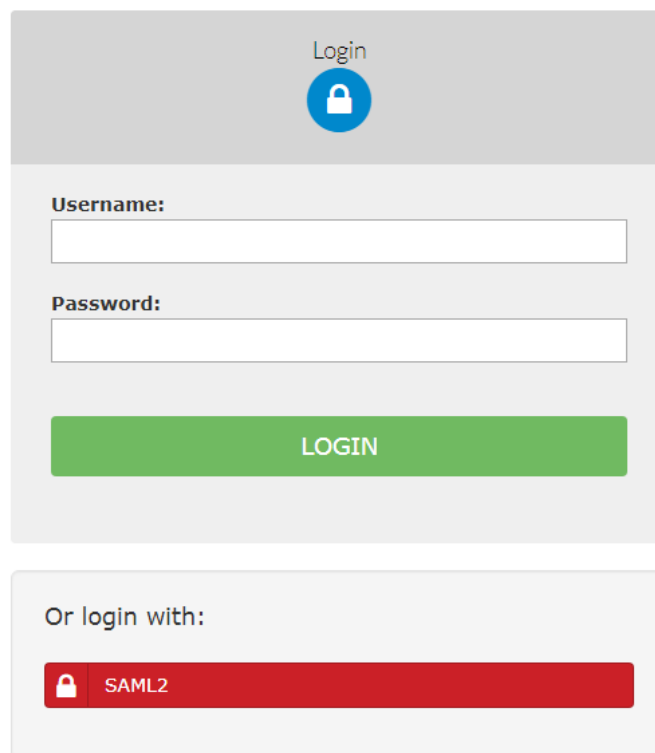
```
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="<URL>" index="<INDEX>" isDefault="<IS_DEFAULT>"/>
```

5.1.2 User Identification in RINA Context

The *userId*, which will be used as a unique user identifier to obtain his profile in RINA is retrieved from the Subject's *NameId* element. That's why the *NameId* should have a "persistent" or "unspecified" *NameFormat* (see chapter "8.3 Name Identifier Format Identifiers" in <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>).

5.2 Workflow

The authentication process starts as described in the [section 3.1](#). However, the user does not provide any username/password on the CAS server login page. Instead, he selects to authenticate with the SAML 2.0 identity provider by clicking the red button as shown on the next figure:



The login form is presented in two main sections. The top section has a grey header with the word 'Login' and a blue padlock icon. Below this, there are two input fields: 'Username:' and 'Password:'. A green 'LOGIN' button is positioned below the password field. The bottom section, separated by a light grey border, is titled 'Or login with:' and contains a red button with a white padlock icon and the text 'SAML2'.

Figure 11. Identity and access management use cases

This redirects the user to the identity provider where he logs in. After a successful log in, the SAML 2.0 assertions are sent back to the CAS server, which creates the TGT and ST for the user and the user continue back to the RINA page, where the CPI module authenticates the ST in the same way as with any other authentication provider.

The workflow with all the RINA components involved is depicted on the Figure 12.

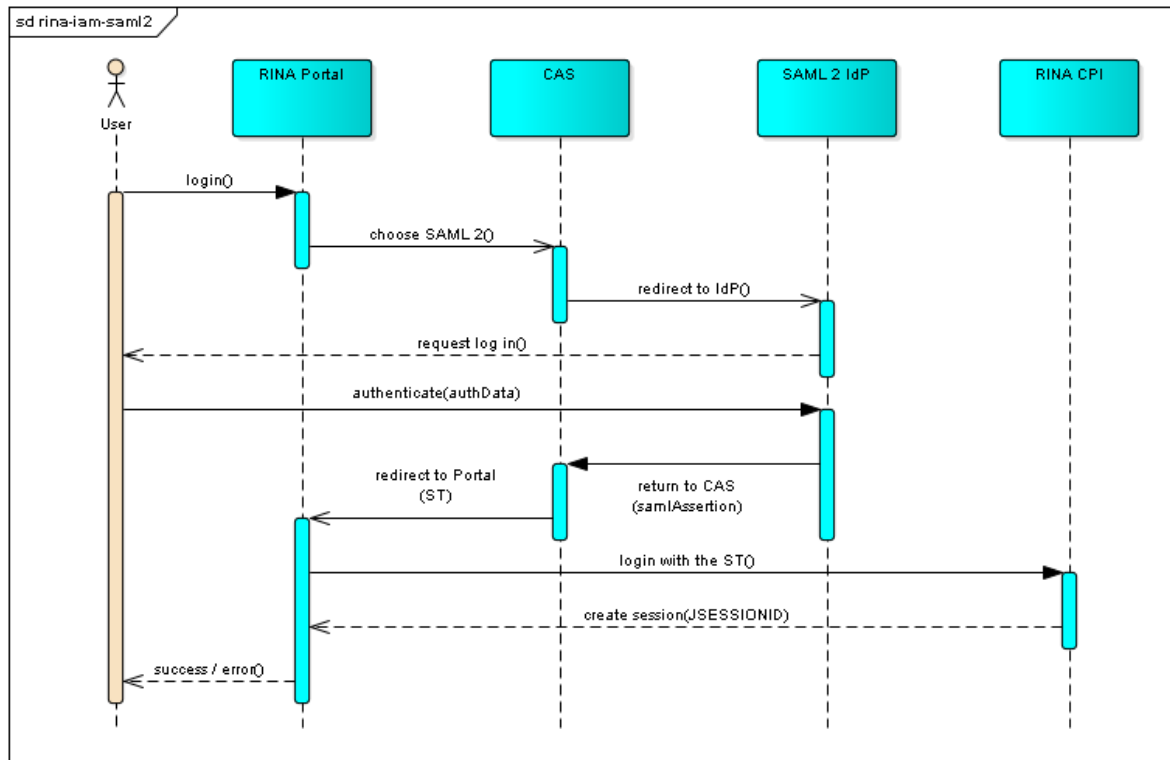


Figure 12. Sequence diagram with delegated SAML 2 IdP

5.3 Configuration

To allow to authenticate against an external SAML 2.0 identity provider, two crucial configurations must be performed:

- Insert the SAML 2.0 settings into the ***cas.properties*** file.
- Import user accounts into RINA IAM datastore with the correct username field.

5.3.1 CAS Configuration

The CAS configuration regarding the SAML 2.0 authentication can be found at

<https://apereo.github.io/cas/6.2.x/integration/Delegate-Authentication.html> and

<https://apereo.github.io/cas/6.2.x/integration/Configuring-SAML-SP-Integrations.html>.

Our provided CAS server is already built with all the needed dependencies. The remaining part is to correctly setup the configuration properties in the ***cas.properties*** file (this file is located usually somewhere in the shared location – consult the RINA installation KIT tutorial to locate the file).

Following is an example SAML 2.0 configuration from the above mentioned file:

```

•
• # SAML 2.0 configuration
• cas.authn.pac4j.saml[0].keystorePassword=demo-passwd
• cas.authn.pac4j.saml[0].privateKeyPassword=demo-passwd
• cas.authn.pac4j.saml[0].serviceProviderEntityId=urn:mace:saml:pac4j.org
• cas.authn.pac4j.saml[0].serviceProviderMetadataPath=file:c:/etc/cas/config/sp-
  metadata.xml
• cas.authn.pac4j.saml[0].keystorePath=file:c:/etc/cas/config/samlKeystore.jks
• cas.authn.pac4j.saml[0].identityProviderMetadataPath=<identity-provider-metadata-url>
• cas.authn.pac4j.saml[0].maximumAuthenticationLifetime=18000
  
```

•

Settings parameter:

- *keystorePassword* – java keystore password of the keystore where some keying material for communication with this identity provider will be stored
- *privateKeyPassword* – password which is used to access the private key
- *serviceProviderEntityId* – entity ID of the service provider, which should also be used in the service provider metadata
- *serviceProviderMetadataPath* – location of the service provider metadata file (typically stored near the CAS configuration file)
- *keystorePath* – location of the keystore (if the keystore does not exist, it will be created). This password of the keystore (and of the primary key) must be the ones specified.
- *maximumAuthenticationLifetime* – lifetime of this authentication in seconds.

Important to notice is, that when using a CAS REST API to obtain the TGT or ST, CAS will not automatically resend any username/password to the delegated identity provider. In this case a custom implementation must be used to handle this.

5.3.2 RINA Configuration

If a delegated SAML 2.0 identity provider is used, the user is authenticated on a different system and CAS will receive the SAML 2.0 assertion. The RINA CPI security module will always get the unique identifier of the user when validating the service ticket.

The user objects must still be present in the RINA database and be able to be found by this unique identification. Because the user objects are identified by the username, this requires the user objects, which are able to log in through a SAML 2.0 identity provider, to have the username equal to this identifier.

CAS provides two ways how this user identifier will be returned:

- Typed - includes a predefined type prefix. This is in case of SAML2.0 following
- Untyped – in this case only the user id will be returned without the type prefix.

- `<the-actual-userid>`
-

This can be controlled by the following configuration setting in the ***cas.properties*** file:

```
# use typed identifier
cas.authn.pac4j.typedIdUsed=true
# use untyped identifier
cas.authn.pac4j.typedIdUsed=false
```

Depending on the configuration used, RINA must contain a user object with the username set to that identifier (either typed or untyped).

6 Configuration for JWT Authentication

To find out more about the CAS configuration when using JWT token for authentication please also read this document:

<https://apereo.github.io/cas/6.2.x/installation/Configure-ServiceTicket-JWT.html>

To enable the JWT authentication following properties need to be defined in the service configuration file configuration file:

```
{
  "@class" : "org.apereo.cas.services.RegexRegisteredService",
  "name" : "EESSI Rina CPI",
  "id" : 101,
  "theme" : "rina",
  "serviceId" : "^(https|http)://localhost.*",
  "attributeReleasePolicy" : {
    "@class" : "org.apereo.cas.services.ReturnAllAttributeReleasePolicy"
  },
  "properties" : {
    "@class" : "java.util.HashMap",
    "jwtSigningSecret" : {
      "@class" : "org.apereo.cas.services.DefaultRegisteredServiceProperty",
      "values" : [ "java.util.HashSet", [ <SIGNATURE_KEY> ] ]
    },
    "jwtEncryptionSecret" : {
      "@class" : "org.apereo.cas.services.DefaultRegisteredServiceProperty",
      "values" : [ "java.util.HashSet", [ <ENCRYPTION_KEY> ] ]
    },
    "jwtSigningSecretAlg" : {
      "@class" : "org.apereo.cas.services.DefaultRegisteredServiceProperty",
      "values" : [ "java.util.HashSet", [ <SIGNING_ALGORITHM> ] ]
    },
    "jwtEncryptionSecretAlg" : {
      "@class" : "org.apereo.cas.services.DefaultRegisteredServiceProperty",
      "values" : [ "java.util.HashSet", [ <ENCRYPTION_ALGORITHM> ] ]
    },
    "jwtEncryptionSecretMethod" : {
      "@class" : "org.apereo.cas.services.DefaultRegisteredServiceProperty",
      "values" : [ "java.util.HashSet", [ <ENCRYPTION_SECRET_METHOD> ] ]
    }
  }
}
```

The JWT token is a self-containing token, which contains the user information. In this kind of authentication, there is no external system connected to the process, but everything is contained within the token. Once this is successfully authenticated a service ticket is generated and the authentication process continues within the CAS 3.0 protocol.

Following is a sequence of HTTP calls (with *CURL*), which can be used to obtain an authenticated session by providing a JWT token instead of username/password.

```
curl -ki -XGET "<SERVER_URL>/eessiCas/login?service=<SERVICE_ID_URL>&token=<JWT_TOKEN>"
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Cache-Control: no-store
Pragma:
Expires:
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Set-Cookie: TGC=<TGC_COOKIE_VALUE>; Path=/eessiCas/; Secure; HttpOnly
Location: <SERVICE_ID_URL>?ticket=<SERVICE_TICKET>
Content-Length: 0
Date: Wed, 28 Feb 2018 09:31:52 GMT
```

With the generated service ticket, you are able to obtain the authenticated session by calling the REST CPI login method.

7 Configuration of Single-Sign-On

The CAS authentication server provides Single-Sign-On(SSO) capabilities. This means, that if the authenticated SSO session (represented by the TGC – ticket-granting-cookie) is still valid, the user does not have to provide credentials again and will be automatically granted a Service Ticket upon accessing CAS login page.

This behaviour is important when changing context of the application. In the case of RINA, we have the following application context:

- Case processing interface (together with the administration part)

A user will receive two separate JSESSION_ID cookies for each context. In order for the user not provide his credentials every time when switching the contexts, the SSO will automatically assign a Service-Ticket to the user if he is already signed in the CAS.

By default, this behaviour is available only if the channel is secure, i.e. SSL/TLS is used on the CAS. If the CAS is deployed without SSL/TLS configured, the user will have to provide his credentials every time he is switching contexts or the session will expiry.

We **STRONGLY** recommend to use SSL/TLS. However, if the SSL/TLS is not used, there is a possibility how to enable the SSO behaviour anyway:

- Set the secure flag on the Tomcat connector to "**true**", where the CAS is deployed:

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
secure="true" />
```
- In the "*cas.properties*" file add the following configuration property:

```
cas.tgc.secure=false
```

Please keep in mind, that this setting can be understood as a security breach, because the TGC cookie is the SSO security context holder. Making this change in the configuration will enable transferring the cookie through an unsecure connection.

8 Other Authentication Plugins

To keep the size of the binary of RINA CAS authentication server reasonably big, only some of the authentication plugins are built in and were tested against a sample environment:

- Delegated authentication (SAML 2.0, OAuth2, OpenID and CAS) – see <https://apereo.github.io/cas/6.2.x/integration/Delegate-Authentication.html>
Token Authentication – see <https://apereo.github.io/cas/6.2.x/installation/Configure-ServiceTicket-JWT.html>

LDAP Authentication – see <https://apereo.github.io/cas/6.2.x/installation/LDAP-Authentication.html>
RINA CAS authentication server supports, however, other authentication mechanisms as well.

8.1 Other Authentication Plugins

The recommended way to install other authentication plugins is to extend the WAR artefact of the RINA CAS Server with the `maven-war-plugin` as a war overlay. This way the developers are free to add additional dependencies or even custom code. For this to work you have to manually install the artefact file into your local repository:

```
mvn install:install-file -Dfile=<path to eessiCas.war>
```

This command (depending on your maven version, please see <https://maven.apache.org/guides/mini/guide-3rd-party-jars-local.html>) will put the `eessiCas.war` artefact into your maven repository. The next step is to create a new maven project with the WAR packaging and include the following into the `pom.xml` file (example is just for demonstration purposes):

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
    4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example.rinacas</groupId>
  <artifactId>custom-cas</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>custom-cas</name>
  <properties>
    <cas.version>6.1.6</cas.version>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <rina.version>6.2.1</rina.version>
  </properties>
  <build>
    <finalName>custom-cas</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.0.3</version>
        <configuration>
          <warName>custom-cas</warName>
          <failOnMissingWebXml>>false</failOnMissingWebXml>
          <recompressZippedFiles>>false</recompressZippedFiles>
          <archive>
            <compress>>false</compress>
          </archive>
          <manifestFile>${project.build.directory}/war/work/eu.ec.dgempl.eessi/eessi-
            cas-server/META-INF/MANIFEST.MF</manifestFile>
          </archive>
          <overlays>
            <overlay>
```

```
•                                     <groupId>eu.ec.dgempl.eessi</groupId>
•                                     <artifactId>eessi-cas-server</artifactId>
•                                     </overlay>
•                                     </overlays>
•                                     </configuration>
•                               </plugin>
•
•                               <plugin>
•                                 <groupId>org.apache.maven.plugins</groupId>
•                                 <artifactId>maven-compiler-plugin</artifactId>
•                                 <version>3.3</version>
•                               </plugin>
•         </plugins>
•     </build>
•
•     <dependencies>
•
•         <dependency>
•             <groupId>eu.ec.dgempl.eessi</groupId>
•             <artifactId>eessi-cas-server</artifactId>
•             <version>${rina.version}</version>
•             <type>war</type>
•             <scope>runtime</scope>
•         </dependency>
•
•         <!-- Other Dependencies -->
•
•     </dependencies>
•
• </project>
```

After building such project, the resulting WAR file can be deployed instead of the one provided. Depending on the name of the resulting deployment and it's context you may need to also change the configuration files of RINA.

To see other authentication plugins and their configuration please visit the following link: <https://apereo.github.io/cas/6.2.x/installation/Configuring-Authentication-Components.html>.

If none of the provided authentication plugins is well suited for your environment, there is also a possibility to implement a custom plugin by yourself. To learn more about the way to implement you custom authentication handler, follow this documentation: <https://apereo.github.io/cas/6.2.x/installation/Configuring-Custom-Authentication.html> . The custom implementation can be included as a source code in the maven WAR overlay as described earlier.

NOTE:

There is no support for the overlaying of the RINA CAS Server module and using other plugins or custom implementations than the ones provided in the binary release. This is due to the fact, that it is not manageable for the EESSI team to test all the plugins and the possible scenarios, nor the custom implementations.

9 Samples

9.1 Update LDAP connection settings request

```
• PUT /Configurations/LdapConnectionSettings
• {
•   "id": "example_id",
•   "version": 1,
•   "institutionId": "institution_name",
•   "providerUrl": "ldap://provider_address:389",
•   "securityAuthentication": "simple",
•   "user": "user@domain.name",
•   "password": "secretPassword",
•   "ldapUserSearchBase": "OU=istitution,DC=domain,DC=suffix",
•   "ldapGroupSearchBase": "OU=istitution_group,DC=domain,DC=suffix",
•   "userSearchString": null,
•   "groupSearchString": null
• }
```

9.2 Get LDAP connection settings response

```
• GET /Configurations/LdapConnectionSettings/{institutionId}
• {
•   "id": "example_id",
•   "version": 1,
•   "institutionId": "institution_name",
•   "providerUrl": "ldap://provider_address:389",
•   "securityAuthentication": "simple",
•   "user": "user@domain.name",
•   "password": "secretPassword",
•   "ldapUserSearchBase": "OU=istitution,DC=domain,DC=suffix",
•   "ldapGroupSearchBase": "OU=istitution_group,DC=domain,DC=suffix",
•   "userSearchString": null,
•   "groupSearchString": null
• }
```

9.3 Update LDAP group parameters mapping request

```
• PUT /Configurations/LdapGroupParametersMapping
• {
•   "institutionId": "institution_name",
•   "ldapUniqueIdentifier": "identifier_attribute",
•   "creationDate": "whencreated",
•   "description": null,
•   "displayName": "cn",
•   "isOrganisationUnit": null,
•   "lastUpdate": null,
•   "name": "cn",
•   "needReassignment": null,
•   "organisationUnit": null,
•   "parentGroupId": null,
•   "members": null,
•   "path": null,
•   "version": 1
• }
```

9.4 Get LDAP group parameters mapping response

```
• GET /Configurations/LdapGroupParametersMapping/{institutionId}
• {
•   "id": "group_id",
•   "institutionId": "institution_name",
•   "ldapUniqueIdentifier": "identifier_attribute",
•   "creationDate": "whencreated",
•   "description": null,
•   "displayName": "cn",
•   "isOrganisationUnit": null,
•   "lastUpdate": null,
•   "name": "cn",
•   "needReassignment": null,
•   "organisationUnit": null,
•   "parentGroupId": null,
•   "members": null,
•   "path": null,
•   "version": 1
• }
```

- }

9.5 Get LDAP group parameters keys

```
• GET /Configurations/LdapGroupParametersKeys
• [
•   "ldapUniqueIdentifier",
•   "name",
•   "displayName",
•   "description",
•   "isOrganisationUnit",
•   "needReassignment",
•   "parentGroupId",
•   "path",
•   "creationDate",
•   "lastUpdate",
•   "members"]
```

9.6 Update LDAP user parameters keys request

```
• PUT /Configurations/LdapUserParametersMapping/{institutionId}
• {
•   "version": 0,
•   "institutionId": "institution name",
•   "ldapUniqueIdentifier": "identifier_attribute",
•   "username": "userPrincipalName",
•   "email": "userPrincipalName",
•   "firstName": "givenName",
•   "lastName": "sn",
•   "creationDate": null,
•   "groups": null,
•   "phoneNumber": null,
•   "isEnabled": null,
•   "isLocked": null,
•   "isAdministrator": null,
•   "lastUpdate": null,
•   "groups": "memberOf",
•   "role": "EVERYONE"
• }
```

9.7 Get LDAP user parameters keys

```
• GET /Configurations/LdapUserParametersMapping/{institutionId}
• {
•   "id": "example_id",
•   "version": 1,
•   "ldapUniqueIdentifier": "dn",
•   "institutionId": "institution id",
•   "username": "userPrincipalName",
•   "email": "userPrincipalName",
•   "firstName": "givenName",
•   "lastName": "sn",
•   "creationDate": null,
•   "groups": null,
•   "phoneNumber": null,
•   "isEnabled": null,
•   "isLocked": null,
•   "isAdministrator": null,
•   "lastUpdate": null,
•   "groups": "memberOf",
•   "role": "EVERYONE"
• }
```

9.8 Get LDAP user parameters keys

```
• GET /Configurations/LdapUserParametersKeys
• [
•   "ldapUniqueIdentifier",
•   "username",
•   "firstName",
•   "lastName",
•   "email",
•   "phoneNumber",
•   "isEnabled",
•   "isLocked",
•   "isAdministrator",
•   "creationDate",
•   "lastUpdate",
•   "groups",
•   "role"
• ]
```

9.9 Synchronize LDAP

request

```
• POST /ApplicationProfile/IAMSettings/LdapSynchronization/{institutionId}
```

response

```
• 17
```

9.10 Reassign group affected cases

request

```
• POST /ApplicationProfile/LdapSynchronization/Reassignment
```

response

```
• None.
```

9.11 Retrieves a list of policies, optionally filtered

Retrieves a list of policies, optionally filtered by policy name or by an associated target.

Request

```
• GET /AssignmentPolicies?institutionId=1&name=POLICY&targetId=P
```

Response

```
• [{
•   id: "policy1",
•   name: "WE vs EE assignment policy",
•   appRole: "CP",
•   rules: [ {
•     id: "rule1",
•     userGroups: [ { id: "101", name: "/Pensions/WE", type: "Group" },
•                   { id: "102", name: "/Sickness/WE", type: "Group" },
•                   { id: "103", name: "/Unemployment/WE", type: "Group" } ],
•     actors: [],
•     condition: {
•       appRole: "CP",
•       ownerCountry: [ "BE", "DE", "FR", "UK" ]
•     }
•   }, {
•     id: "rule2",
•     userGroups: [ { id: "201", name: "/Pensions/EE", type: "Group" },
•                   { id: "202", name: "/Sickness/EE", type: "Group" },
•                   { id: "203", name: "/Unemployment/EE", type: "Group" } ],
•     actors: [ "Supervisor", "Authorized", "Nonauthorized", "Viewer",
•               "Auditor" ],
•   }
• ]
```

```
•         condition: {
•             appRole: "CP",
•             ownerCountry: [ "BG", "RO", "HU", "SK" ]
•         }
•     ]
• },
• {
•     id: "policy2",
•     name: "Assignment by sector policy ",
•     appRole: "CP",
•     rules: [{
•         id: "rule3",
•         userGroups: [ { id: "10", name: "/Sickness", type: "Group" } ],
•         actors: "actors" : ["Auditor", "Authorized", "Medical",
"Nonauthorized", "Supervisor", "Viewer", "Vip"],,
•         condition: {
•             appRole: "CP",
•             sector: [ "S" ]
•         }
•     }, {
•         id: "rule4",
•         userGroups: [ { id: "11", name: "/Pensions", type: "Group" } ],
•         actors: [],
•         condition: {
•             appRole: "CP",
•             sector: [ "P" ]
•         }
•     }, {
•         id: "rule5",
•         userGroups: [ { id: "12", name: "/Unemployment", type: "Group" } ],
•         actors: [ ],
•         condition: {
•             appRole: "CP",
•             sector: [ "UB" ]
•         }
•     }
• ]
• }}]
```

9.12Retrieves a policy

Request

```
• GET /AssignmentPolicies/policy2?institutionId=1
```

Response

```
• {
•     id: "policy2",
•     name: "Assignment by sector policy ",
•     appRole: "CP",
•     rules: [{
•         id: "rule3",
•         userGroups: [ { id: "10", name: "/Sickness", type: "Group" } ],
•         actors: [],
•         condition: {
•             appRole: "CP",
•             sector: [ "S" ]
•         }
•     }, {
•         id: "rule4",
•         userGroups: [ { id: "11", name: "/Pensions", type: "Group" } ],
•         actors: [],
•         condition: {
•             appRole: "CP",
•             sector: [ "P" ]
•         }
•     }, {
•         id: "rule5",
•         userGroups: [ { id: "12", name: "/Unemployment", type: "Group" } ],
•         actors: [ ],
•         condition: {
•             appRole: "CP",
•             sector: [ "UB" ]
•         }
•     }
• ]
• }
```

```
•      }  
•      }  
•    ]  
•  }
```

9.13 Creates a policy

Request

```
• POST /AssignmentPolicies?institutionId=1  
• {  
•   name: "WE vs EE assignment policy",  
•   appRole: "CP",  
•   rules: [ {  
•     id: "rule1",  
•     userGroups: [ { id: "101", name: "/Pensions/WE", type: "Group" },  
•                   { id: "102", name: "/Sickness/WE", type: "Group" },  
•                   { id: "103", name: "/Unemployment/WE", type: "Group" },  
•     actors: ["Auditor", "Authorized", "Medical", "Nonauthorized",  
• "Supervisor", "Viewer", "Vip"],  
•     condition: {  
•       appRole: "CP",  
•       ownerCountry: [ "BE", "DE", "FR", "UK" ]  
•     }  
•   }, {  
•     id: "rule2",  
•     userGroups: [ { id: "201", name: "/Pensions/EE", type: "Group" },  
•                   { id: "202", name: "/Sickness/EE", type: "Group" },  
•                   { id: "203", name: "/Unemployment/EE", type: "Group" },  
•     actors: [],  
•     condition: {  
•       appRole: "CP",  
•       ownerCountry: [ "BG", "RO", "HU", "SK" ]  
•     }  
•   }  
• ]  
• }
```

Response

```
• policy1
```

9.14 Updates a policy

Request

```
• PUT /AssignmentPolicies/policy2?institutionId=1  
• {  
•   id: "policy2",  
•   name: "Assignment by sector policy",  
•   appRole: "CP",  
•   rules: [{  
•     id: "rule3",  
•     userGroups: [ { id: "10", name: "/Sickness", type: "Group" } ],  
•     actors: [],  
•     condition: {  
•       appRole: "CP",  
•       sector: [ "S" ]  
•     }  
•   }, {  
•     id: "rule4",  
•     userGroups: [ { id: "11", name: "/Pensions", type: "Group" } ],  
•     actors: [],  
•     condition: {  
•       appRole: "CP",  
•       sector: [ "P" ]  
•     }  
•   }, {  
•     id: "rule5",  
•     userGroups: [ { id: "12", name: "/Unemployment", type: "Group" } ],  
•     actors: [],  
•     condition: {  
•       appRole: "CP",  
•       sector: [ "U" ]  
•     }  
•   }  
• ]  
• }
```

```
•                                     appRole: "CP",  
•                                     sector: [ "UB" ]  
•                                     }  
•                                     ]  
•                                     }  
•                                     ]  
•                                     }
```

Response

- None

9.15 Deletes a policy

Request

- DELETE /AssignmentPolicies/policy1?institutionId=1

Response

- None

9.16 Associates an existing policy to a target

Request

- PUT /AssignmentPolicies/policy2/Targets/PO_S_BUC_01?institutionId=1

Response

- None

9.17 Associates existing policies to a target

Request

- PUT /AssignmentPolicies/Target/PO_S_BUC_01?institutionId=1
["acc6e419738b4ec39e969a7eb51827f7", "ae0b1256186e4c1aaa384e03de9383aa"]

Response

- None

9.18 Removes a target from a policy

Request

- DELETE /AssignmentPolicies/policy2/Targets/PO_S_BUC_01?institutionId=1

Response

- None

10PDP - Case Assignment Policy Decision Point

The case assignment policy decision point is called once for every new case instance.

The input is an object representing the case instance. The output is the list of case assignments.

Based on the case instance properties, Case Assignment Policy Decision Point determines the applicable policies, evaluates them and computes the list of case assignments for this case (a list of groups and/or users for each actor).

10.1 Service contracts / interface

```
• interface CaseAssignmentsPDP {  
•     List<Actor> calculateCaseAssignments(String institutionId, CaseProperties caseProperties)  
•         throws InvalidCaseException, NoApplicablePolicyException,  
•             PolicyProcessingException;  
• }
```

10.2 Data contracts / model

10.2.1 Actor

The actor of a case

Attribute	Description	Data Type
id	The id of the actor	String
name	The name of the actor	String
userGroups	A list of users or groups that will be assigned to the case as this actor	List<UserOrGroup>

10.2.2 UserOrGroup

Represents a user or a group.

Attribute	Description	Data Type
id	The id of this user/group in RINA. Specific constants are used to represent the <Creator's Group> and <Creator User> placeholders	String
name	The name of "User" or "Group"	String
type	Selection of "User" or "Group"	String

10.2.3 Case properties

Represents a new case instance.

Attribute	Description	Data Type
id	the id of the case	String
processDefinitionName	the identifier of the process of this case (e.g. CP_P_BUC_07)	String
participants	the list of participants to this case	List<Participant>

subject	the person or organization that this case is about	Entity
creator	the creator user of this case	UserOrGroup

10.2.4 Participant

Attribute	Description	Data Type
role	The application role of this participant (PO/CP)	String
organisation	the participant organization	Organisation

10.2.5 Organisation (inherits Entity)

Attribute	Description	Data Type
id	The id of the organisation	String
name	The organisation name	String
acronym	The acronym for the organisation	String
countryCode	The Country code	String

10.2.6 Person (inherits Entity)

Attribute	Description	Data Type
pid	The id of the person	String
name	The name of the person	String
surname	The surname of the person	String
birthday	The birthday of the person	Date
sex	The sex of the person	String

10.2.7 Entity

Attribute	Description	Data Type
address	The address of the entity	Address

10.2.8 Address

Attribute	Description	Data Type
street	The street of the entity	String
town	The town of the entity	String
postalCode	The postalCode of the entity	String
region	The region of the entity	String
country	The country of the entity	String

10.3 Fault contracts / exceptions

```
InvalidCaseException: { // something is missing or invalid in the input
}
NoApplicablePolicyException: { // could not find an applicable policy
}
PolicyProcessingException: { // something else went wrong when evaluating policies
}
```

11 Development impact

Changing the authentication way is going to imply that an interface is needed to select which one of Internal RINA or LDAP is preferred, coming with a few constraints:

- IAM Related:
 - Using an external DB like LDAP to store the credentials (and the authorizations) means that RINA *depends* on LDAP. Therefore, it is not possible to create/update/delete users or groups from the application (through the use of LDAP); However, the default role assigned during synchronization can be updated and other group memberships can be added.
 - All the cases' assignments must be updated when the group changes, so a tool must be created to accommodate this synchronization;
 - Add user to multiple groups support as currently the users cannot belong to more than one group;
 - Add membership (role of a user inside a group).
- IAM Related and External Rules: Two types of rules: process owner and counterparty based.

12How-to's

12.1How to configure the synchronizer

See **Error! Reference source not found.** for the fields' descriptions

12.2How to develop my external rules

The basic structure of a rule is: "Assign <Groups or Users> as <Actors> if <Conditions>".

In the editing screen, fill-in all fields:

- sector / process
- role
- actor
- groups (e.g.: sending country). assigned users/groups

All fields left blank are not considered, therefore it means "any".

A common scenario would be having inside an organization specific groups of users to be assigned to every sector, like the groups in this image:

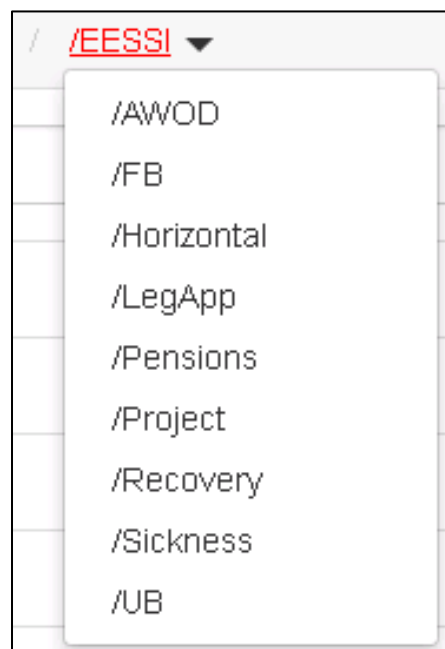


Figure 13. Illustrative example on having specific groups of users

To address this scenario, a single assignment policy like the one described below can be created and then assigned to all BUC sectors:

The policy would contain several assigning rules, each rule linking one of the sectors to the corresponding user group (e.g. the "P" sector to the "/EESSI/Pensions" group):

```
{
  "name": "Assign EESSI subgroups by sector",
  "description": "This policy assigns every case to the corresponding subgroup based on the sector (Pension, Sickness etc)",
  "color": "green",
  "type": "POLICY",
  "rules": [{
    "userGroups": [{
      "name": "/EESSI/AWOD",
      "id": "101",
```

```
        "type": "Group"
      },
      "actors": [],
      "condition": {
        "sector": ["AW"]
      }
    }, {
      "userGroups": [{
        "name": "/EESSI/FB",
        "id": "102",
        "type": "Group"
      }
    ],
    "actors": [],
    "condition": {
      "sector": ["FB"]
    }
  }, {
    "userGroups": [{
      "name": "/EESSI/Horizontal",
      "id": "103",
      "type": "Group"
    }
  ],
  "actors": [],
  "condition": {
    "sector": ["H"]
  }
}, {
  "userGroups": [{
    "name": "/EESSI/LegApp",
    "id": "104",
    "type": "Group"
  }
],
"actors": [],
"condition": {
  "sector": ["LA"]
}
}, {
  "userGroups": [{
    "name": "/EESSI/Pensions",
    "id": "105",
    "type": "Group"
  }
],
"actors": [],
"condition": {
  "sector": ["P"]
}
}, {
  "userGroups": [{
    "name": "/EESSI/Recovery",
    "id": "107",
    "type": "Group"
  }
],
"actors": [],
"condition": {
  "sector": ["R"]
}
}, {
  "userGroups": [{
    "name": "/EESSI/Sickness",
    "id": "108",
    "type": "Group"
  }
],
"actors": [],
"condition": {
  "sector": ["S"]
}
}, {
  "userGroups": [{
    "name": "/EESSI/UB",
    "id": "109",
```

```
        "type": "Group"
      }
    ],
    "actors": [],
    "condition": {
      "sector": ["UB"]
    }
  }
]
```

12.3 How to configure LDAP

12.3.1 Authentication

```
cas.authn.ldap[0].ldapUrl=ldap://ldapServerIpAddress:389
cas.authn.ldap[0].useSsl=false
cas.authn.ldap[0].allowMultipleDns=true
cas.authn.ldap[0].baseDn=OU=instName,DC=rinaldp,DC=lab
cas.authn.ldap[0].userFilter=(&(samAccountName={user}))(objectClass=user)
cas.authn.ldap[0].subtreeSearch=true
cas.authn.ldap[0].bindDn=install@rinaldp.lab
cas.authn.ldap[0].bindCredential=pass123
```

13 Relevant links & References

The following links have been used to create this specification and are useful for development:

- LDAP Community milestones for supporting LDAP

<https://www.openldap.org/>

- LDAP Authentication
- XACML - eXtensible Access Control Markup Language (XACML) Version 3.0

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>

14 Limitations of the current version

In the current version the RINA external rules interface is not yet fully defined. Therefore, in the next versions of this document, for the external rules interface the following may be defined:

- Service contract
 - This will define the operations, which are exposed by the RINA external rules interface service to the exterior. The service contract will describe what external rules interface APIs and relevant details (i.e. URI, description, etc.);
- Data contract
 - This will define the data class models of the information that will be exchange between the CPS and the RINA external rules interface service;
- Fault contract
 - This provides documented view for error accorded in the RINA external rules interface service to client. Due to the error masking, as the service throws any error which will not reach CPS, the fault contract will help to easy identify what error has occurred, and where.

References:

"EESSI – RINA Case Processing Interface (CPI) – Reference"