



EESSI - RINA

Case Processing Interface (CPI)

Solution / Application Architecture
EESSI-2020 / rev01

Table of Contents

1	Introduction	7
2	EESSI Case Processing Interface Overview	8
3	Common API Concepts and Features	10
3.1	Multitenancy	10
3.1.1	Description	10
3.1.2	Implementation Approach	11
3.2	Date format	12
3.3	User Authentication	12
3.3.1	Ticket-granting Ticket Request	14
3.3.2	Ticket-granting Ticket Response	14
3.3.3	Service Ticket Request	15
3.3.4	Service Ticket Response	15
3.3.5	Logging in with the Service Ticket	16
3.3.6	Validating a Service Ticket	17
3.4	CSRF Protection	18
3.5	CORS Filtering	20
3.6	Authorisation to access protected resources	20
3.7	Data Validation	24
3.7.1	Data validation at SED level	24
3.7.2	Data validation at message level	25
3.8	Audit trail	25
3.8.1	RINA Audit Trail Architecture	25
3.8.2	Audit Trail Persistence & Retrieval	26
3.8.3	General approach to audit trail data model	26
3.8.4	Particular approach of audit trailing for Case Processing Interface	40
3.9	HTTP Error codes	42
4	Functions and use cases	44
4.1	Search management function	45
4.1.1	Run search	45
4.1.2	Manage predefined search	45
4.1.3	Run predefined search	46
4.1.4	Run refined predefined search	46
4.1.5	Configure Search Result	46
4.1.6	TsVector implementation details	47
4.2	Case management function	48
4.2.1	Create case	48
4.2.2	Read actions	49
4.2.3	Run action	49
4.2.4	Manage document/subdocument attachments	51
4.2.5	Manage document comments	52
4.2.6	Case assignment	52
4.2.7	Manage case comments	52
4.2.8	Manage case attachments	53
4.3	Notification Management Function	53
4.3.1	Notification Centre (Configuration Management)	53
4.3.2	Notification Management	53
4.3.3	Filter notifications	54
4.3.4	Mark notification as read/unread	55
4.3.5	Summarize notifications	55
4.4	Alarms management function	55
4.5	Application Profile management function	55

4.6	Application resources management function	56
4.6.1	Retrieve Tenants' attributes.....	56
4.6.2	Retrieve Tenant attributes	56
4.6.3	Delete Tenant	56
4.6.4	Update Tenant	56
4.7	Assignment policies management function.....	57
4.8	Audit logs management function	57
4.9	Pending Messages and Business Exceptions	57
4.10	Sanity test management function (buckets definitions)	58
4.11	Check Definitions	58
4.12	Entities management function	58
4.13	Files	58
4.14	Keystores management function	58
4.15	Authorisation management function (process definitions assignment)	58
4.16	Physical artefact management function (resources)	58
4.17	Sectors management function.....	59
4.18	Synchronisations management function	59
4.19	Technical logs management function	59
4.20	User Management Function (Groups)	59
4.21	User profile management function	59
4.22	Vocabularies management function	59
4.23	Case Management Scheduling/ Planning.....	59
4.24	Configurations	60
4.25	Portal content function	60
5	Subscription Service (WebSocket).....	61
5.1	Sample Scenario	61
5.2	Notification format.....	61
5.3	Subscribing to the case notifications	62
6	Behaviour - state machines	64
6.1	Case state machine	64
6.2	Document sender state machine	64
6.3	Document receiver state machine.....	65
7	Reference Documentation	66
ANNEX - Differences of CPI Implementation between RINA 5.x (EESSI 2019) and 6.x (EESSI 2020).....		68

Table of Tables

Table 1: Changes in Document Controller of CPI API	69
Table 2: Changes in Cases Controller of CPI API	70
Table 3: Changes in Activity Controller of CPI API	71
Table 4: Changes in ApplicationProfile Controller of CPI API	71
Table 5: Changes in Groups Controller of CPI API	71
Table 6: Changes in Keystore Controller of CPI API	71
Table 7: Changes in LetterTemplates Controller of CPI API	72
Table 8: Changes in Notifications Controller of CPI API	72
Table 9: Changes in ResourceAdmin Controller of CPI API	72
Table 10: Changes in Subscriptions Controller of CPI API	72
Table 11: Changes in Document Ticket Controller of CPI API	73
Table 12: Changes in Users Controller of CPI API	73
Table 13: Changes in ApListener Controller of CPI API	73
Table 14: Changes in authentication Controller of CPI API	74
Table 15: Changes in Organisation Controller of CPI API	74
Table 16: Changes in PortalForms Controller of CPI API	74

Document Control Information

Document Control	Value
Project Title	Electronic Exchange of Social Security Information (EESSI)
Document Name	EESSI – RINA - Case Processing Interface (CPI)
Document Category	Solution / Application Architecture
Revision (date)	rev01 (01/07/2021)
Component	RINA
Last Publication Date Project Milestone	21/12/2020 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

Project Milestone/ Released Component version	Changes/Corrections Description
EESSI 2019 / RINA 5.6.2	Include Medical data and International (Global) ID and Business ID implementation
	Improved content under the Case Management function section
	Added information about Swagger
EESSI 2020 / RINA 6.2.1	Description updated to reflect then new RINA developments and architectural improvements
	Improved search functionality using Postgres TsVector
	Section 1: Updated with the NEW RINA architecture diagram and description
	Section 2.8: updated with the new Audit Trail Architecture
	Section 2.10: Technical and Exception logging eliminated (Information included in the New RINA Architecture Overview document)
	Section 5: updated with new state transition diagrams
EESSI 2020 (rev01)	Minor adjustments in content of section 2.8 for alignment with other Architecture documents (RINA Architecture Overview, RINA Business Messaging Interface).
	Minor clarifications in section 5.1.
	Update of the Service URL in Section "3.3.3 - Service Ticket Request"
	Describe all the detected differences between the CPI implementation of RINA 5.x (EESSI 2019) and RINA 6.x (EESSI 2020). The detailed description of the differences and the backwards compatibility information are given in the ANNEX at the end of the document.

1 Introduction

This document provides a detailed technical description of the Case Processing Interface (CPI) of RINA.

The structure of the document follows:

- Chapter 1: Introduction
- Chapter 2: An overview of the EESSI CPI is given
- Chapter 3: The API concepts and features are presented and described in detail
- Chapter 4: The specifications of the functions and the Use Cases are described per function (i.e., Search management, Case management, Notification management, Authorisation management, Synchronisation management, etc.)
- Chapter 5: the chapter is devoted in the description of the subscription service
- Chapter 6: the necessary state machines that describe the system behavior are described
- Chapter 7: The Reference Documentation is described
- ANNEX: the differences in the implementation between RINA 5.x (EESSI 2019) and RINA 6.x (EESSI 2020 delivery) are described in detail per controller and method.

2 EESSI Case Processing Interface Overview

Part of the National Institutions Domain, RINA (Reference Implementation for National Applications) consists in 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 organisations the tools to implement the International Protocol of Data Exchange based on Structured Electronic Documents in Social Security belonging to European Community Member States and Associated States.

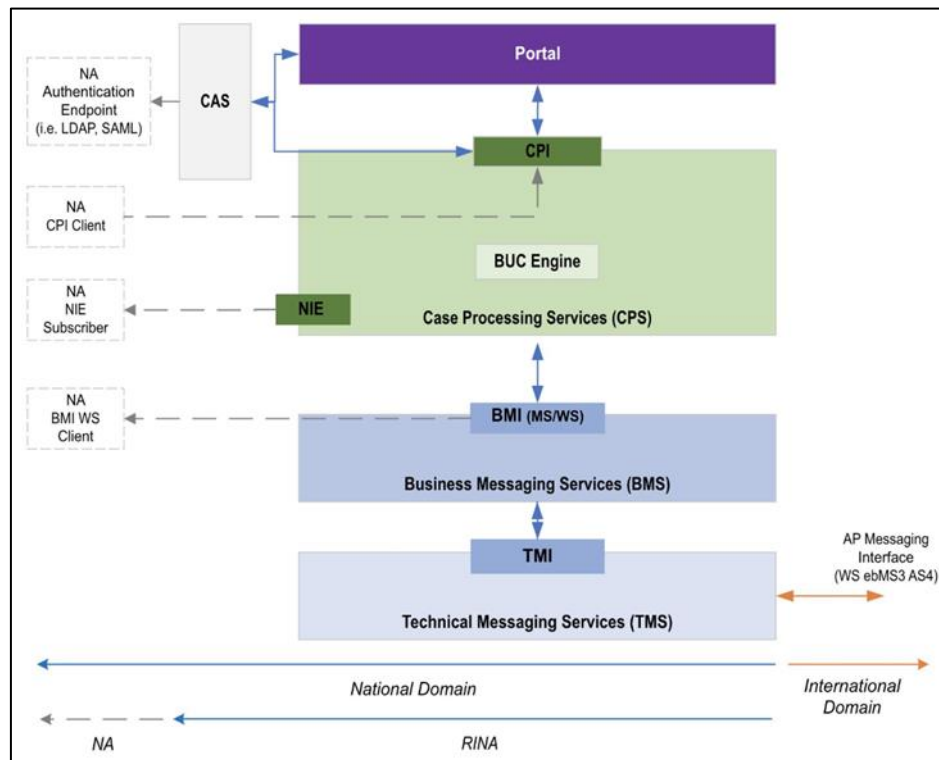


Figure 1 - Overview of RINA interfaces and layers

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

The scope of this document is to provide a specification document for the Case Processing Interface, part of the Case Processing Services layer.

Here is a short description of the document structure:

- Interface overview – this chapter is an introduction to what this interface is about, its place and why it is needed;
- Common API Concepts and Features – an introduction of the API concepts covering the following: user authentication and authorisation, data validation, audit trail, exception management and technical and exception logging;
- Functions and use cases – a presentation of high level functions and use cases that drive this service component;
- Push Notifications;
- Behaviour – a detailed case state machine which is used to keep track of the case state on both the sender the receiver side;
- Reference Documentation – the data contract consisting in the REST logical data model, in parallel to the service contract used for the API.

This interface, previously defined as the "ICD3 like interface", provides a RESTful web service that allows the RINA portal or national systems to interact with the case processing engine.

It provides access to different services like the case Processing service, the guidance service, the document management service or the notification service.

Clients that use this API can manage cases by creating, updating, deleting or retrieving cases. They can also manage the documents belonging to cases, publish attachments and work with comments at both case and document level.

Users can query for notifications if they want to see the outstanding cases and group them by different categories.

Additional details related to the Case Processing Interface functions and use cases, service contract, data contract and relevant samples can be found in other RINA related specifications (RINA Identity and Access Management (IAM), RINA National Information Exchange Interface (NIE), RINA Archiving, RINA Localisation)

3 Common API Concepts and Features

All the REST APIs have some concepts in common that are useful to know and understand before starting to use a particular API.

For example, all APIs have a server base URL, the CRUD operations use the REST approach where they use a uniform interface based on the HTTP verb used and in case of an error the response comes in a standard way.

When performing an API call you need to get authenticated and authorized.

Also, all the important operations are going to be logged and for tracking down specific events we provide audit trail logs.

3.1 Multitenancy

3.1.1 Description

RINA Application can support multiple Institutions that are installed as different Tenants to the system.

Multitenancy is currently implemented only on the Portal level and not on CPI Interface.

RINA Portal consists of two Consoles.

- Administration Console.
- Case Management Console

Administration Console enables the administration of both global and tenant specific configuration.

The Case Management Console enables the case management functionalities independently and transparent to each different Tenant.

3.1.1.1 Administration Console

All Tenants in the same RINA Application share the following common application configuration and administration.

- General Application Settings:
Application Id, Languages, Max Idle Time, Organisation System Password, SED Validation Mode, SED Client Validation, Relay Server Settings, Attachment Settings, Bulk Document Settings, Default User Profile and Localisation Settings are shared.
- Messaging Settings (partially)
BMP Validation Mode, Security Keystore Settings, Transport Mode, Max Message Size, Retry Interval, Maximum Retries, Antimalware Settings,
Access Point Business and System endpoint are shared
- Nie Settings
- RINA Archiving
- Notification Center
- Automatic Updates

- Test Center
- Business Exceptions

Each Tenant Institution has its own independent configuration for the following elements

- IAM
 - Users and Groups
 - Authorisation Policies and Process Assignments
 - LDAP Configuration.
- Case Counter Settings
- Local Messaging Settings
 - Default Business Signature Alias
 - Business and System Endpoint MPC, Pulling Interval
 - Authorization Key Alias Mapping And Password

3.1.1.2 Case Management Console

Search Cases, Case Management, Notifications, Calendar are independent for each Tenant Institution.

Processes instances are different and no interaction between them happens.

3.1.2 Implementation Approach

As mentioned before, the scope of Multitenancy is addressed only on the Portal level. Nevertheless, to enable this changes in other layers were necessary.

CPI Service Contract is only modified to enable the Administration Console to handle the Administration of Tenants.

CPI Service Implementation is also modified, but it still uses the same singleton services for all the Tenants.

Multitenancy is accomplished by distinguishing different data entries for each Tenant when this is applicable. The storage of the data in Postgresql Database is performed in the same indexes and types for all Tenants but in different entries (documents). The Data Model Objects that are tenant specific contain a field that is the reference to the tenantId (InstitutionId) of the Tenant, or a reference to another object that has the reference in the tenantId.

Case Processing Related CPI Services identify which Tenant is performing API Calls, by the user session, since each user belongs only to one Tenant Institution.

More details regarding CPI can be found also in the rest of the document.

Regarding Messaging (and BMI), Multitenancy is accomplished by creating different Pmodes for each Tenant, with the constraint that the Institutions/Tenants of the same Application share the same mode (Pull or Push) and are under the same Access Point.

More details regarding BMI can be found in the BMI Documentation.

3.2 Date format

When communicating with the REST API, the only supported date-time formats are the following:

- yyyy-MMdd'T'HH:mm:ss.SSSX
- yyyy-MM-dd'T'HH:mm:ssX

3.3 User Authentication

User authentication allows a user to login into the case processing system by providing its credentials: username and password. In RINA, authentication mechanism is provided by the Central Authentication Server (CAS) which can be deployed either in the same server as the RINA CPI module or separately.

Each Tenant has its own users (and groups). They are independent to each other with the constraint that the username of a user is unique per RINA Application and not per Tenant.

On the contrary admin user is common between the Tenants. There is a restriction, that an administration user cannot delete his own account. This way, there is always at least one administration user.

The CAS Server is a single component in the RINA Application and in general user Authentication mechanism is shared by all Tenants.

Once the login is granted the Login API will create an authenticated session and return a JSESSIONID cookie. This cookie must be sent with every further request to the CPI. Figure 1 shows the general flow for user authentication, from the moment the client initiates the

login until it receives the authenticated session cookie (the actual paths and URLs may differ based on the deployment).

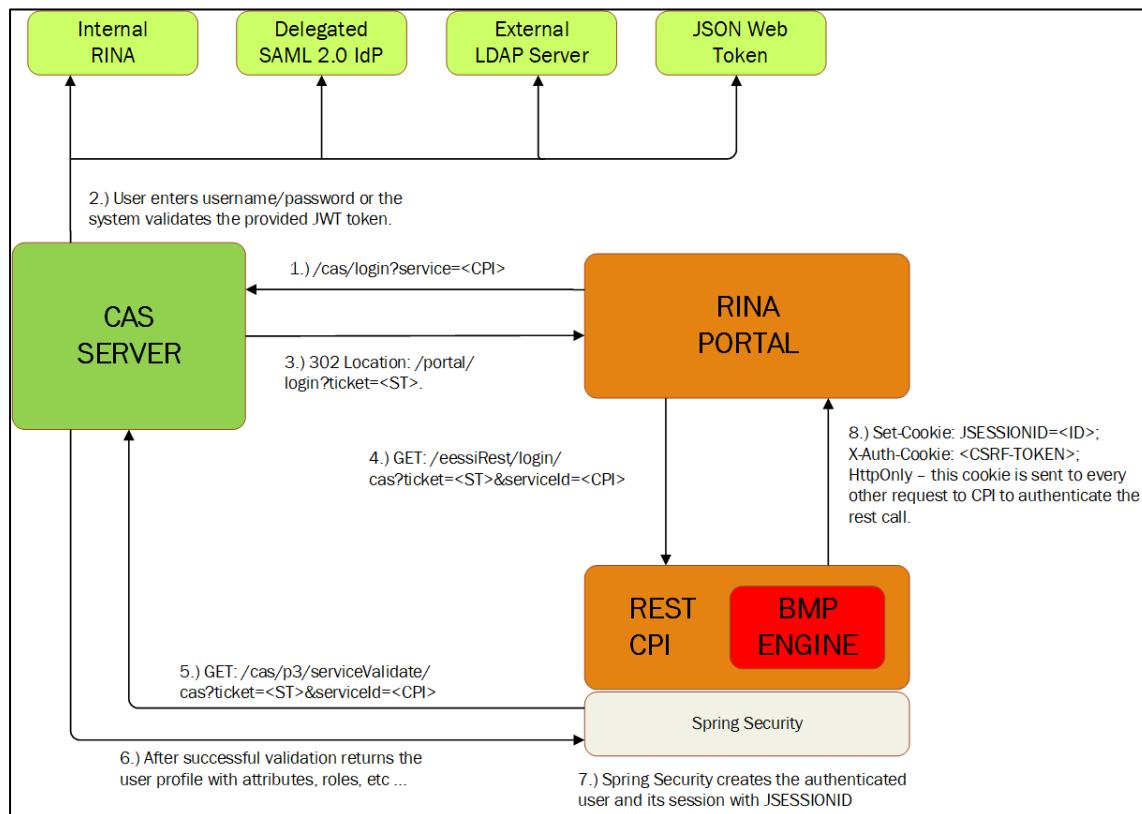


Figure 2. User authentication

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 server providing the authentication to users requesting to use a service.
- Service – CAS is providing the authentication for registered services. In our case, the registered service is the RINA REST CPI module.
- Portal/UI – is the client, which wants to use the service and needs to get authenticated by the CAS server.

To understand the interaction of the parties, we first need to define a couple of terms from the CAS 3.0 protocol. The protocol can be viewed similar to OAuth 2.0. Instead of token we will be speaking here about tickets. CAS issues 2 types of tickets in 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) for authorizations to use a service.
- 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 used only once. Any ST, which has already been validated once, would fail in another validation.

After successfully obtaining a service ticket, this must be validated by the service's authentication module and if it is valid the authenticated and authorized session is created.

The description of the authentication process is given below as clearly stated on Figure 1 (the Steps below correspond to the step shown on the figure):

- **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 via CPI Authentication Endpoint, LDAP, SAML 2.0). If the user is authenticated successfully, a SSO session in the CAS server is created.
- **Step 3:** The CAS server generates a service ticket (ST) for the service ID sent in the request in the step 1. The user is redirected back to the client with the generated service ticket.
- **Step 4:** The client calls a "login" method of the RINA CPI with the service ticket.
- **Step 5:** The RINA CPI security module authenticates the received service ticket by calling the CAS server API method.
- **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 finds 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 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 separated one. The maven project contains the custom internal authentication module and accesses the RINA datastore through a dependency on the RINA services.

In the following chapters we describe the calls, which are provided by the CAS server REST API for obtaining and validating tickets.

3.3.1 Ticket-granting Ticket Request

To obtain a TGT a HTTP POST request must be sent to the CAS server with the following parameters:

- username - REQUIRED. The user's username.
- password - REQUIRED. The user's password.

For example, if the host is server.example.com and the port 8080, the client must make the following HTTP request:

```
Request URL: http://server.example.com:8080/eessiCas/v1/tickets  
username=johndoe&password=my-password  
Request Method: POST  
Request Headers:  
Content-Type: application/x-www-form-urlencoded
```

3.3.2 Ticket-granting Ticket Response

If the authentication was successful a TGT is generated and returned in the Location HTTP header:

HTTP/1.1 201 Created

Location: <http://server.example.com:8080/eessiCas/v1/tickets/TGT-6-JWbzCAGLgVykiI0As43xB5ZEaaXh562L0Ms5WhVPextVl0FiMl-CSD-04>

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"><html><head><title>201
Created</title></head><body><h1>TGT Created</h1><form action="https://localhost:8443/eessi-cas-
server/v1/tickets/TGT-6-JWbzCAGLgVykiI0As43xB5ZEaaXh562L0Ms5WhVPextVl0FiMl-CSD-04"
method="POST">Service:<input type="text" name="service" value=""><br><input type="submit"
value="Submit"></form></body></html>
```

If the authentication has not been successful, a HTTP 401 Unauthorized response is returned:

```
401 Unauthorized
Server: Apache-Coyote/1.1
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Type: text/plain; charset=UTF-8
Content-Length: 64
Date: Wed, 13 Dec 2017 13:24:58 GMT
```

```
{
  "authentication_exceptions" : [ "FailedLoginException" ]
}
```

3.3.3 Service Ticket Request

With a successfully generated TGT, we can request a service ticket to be able to log in to the RINA CPI. To obtain a ST, a HTTP POST request must be sent to the CAS server with the following parameters:

- TGT - REQUIRED. The TGT generated by the previous call.
- service - REQUIRED. The service ID, for which we are requesting the ST

For example, if the host is server.example.com and the port 8080, the client must make the following HTTP request:

```
Request URL: http://server.example.com:8080/eessiCas/v1/tickets/{TGT}
service=http://localhost/cas/cpi
Request Method: POST
Request Headers:
Content-Type: application/x-www-form-urlencoded
```

NOTE: A successfully generated ST is valid for only 10 seconds and can be validated only once.

3.3.4 Service Ticket Response

If the generation of the ST was successful the response will contain the ST:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Type: text/plain; charset=UTF-8
Content-Length: 32
Date: Wed, 13 Dec 2017 13:28:05 GMT

ST-7-TRaZGtOgglxrAaYkqaZB-CSD-04
```

If there was an error during generation of the ST, the HTTP response code will be 500:

```
HTTP/1.1 500 Internal Server Error
Server: Apache-Coyote/1.1
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Transfer-Encoding: chunked
Date: Wed, 13 Dec 2017 13:32:49 GMT
Connection: close
```

3.3.5 Logging in with the Service Ticket

If the user has successfully obtained the ST, he must login with this ticket to RINA CPI module. The security module in CPI has built-in validation of the ST against the CAS server. An example HTTP request:

```
Request URL: http://server.example.com:8080/eessiRest/login/cas?ticket=ST-7-
TRaZGtOgglxrAaYkqaZB-CSD-04&serviceId=http%3A%2F%2Fserver.example.com%3A8080%2FeessiCpi
Request Method: GET
```

In case of successful ST validation, the response contains the JSON representation of the logged-in user and the JSESSIONID cookie. This JSESSIONID cookie must be then sent with every further RINA CPI request:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: PUT, POST, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: content-type, x-requested-with, accept, request-id, origin,
location, authorization
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: JSESSIONID=A1D904D6A6F7C416F37CDED4C1DC217E.server1; Path=/eessi-rina-cpi/; HttpOnly
Content-Length: 1013
```

Date: Wed, 13 Dec 2017 13:41:41 GMT

```
{
  "id": "1", "version": 12, "username": "supervisor", "institutionId": "LOOPBACK", "firstName": "John", "lastName": "Doe", "creationDate": "2017-08-23T11:57:22Z", "password": null, "phoneNumber": null, "email": "supervisor@example.com", "memberships": [
    {
      "group": {
        "id": "1", "version": 9, "name": "ACME", "institutionId": "LOOPBACK", "isOrganisationUnit": false, "needReassignment": false, "displayName": "ACME", "description": "Test group of institution LOOPBACK", "creationDate": "2017-08-23T11:57:22Z", "parentGroupId": "", "path": null, "parentPath": "", "organisationUnit": false
      }, "role": "SUPERVISOR"
    }, {
      "group": {
        "id": "1", "version": 9, "name": "ACME", "institutionId": "LOOPBACK", "isOrganisationUnit": false, "needReassignment": false, "displayName": "ACME", "description": "Test group of institution LOOPBACK", "creationDate": "2017-08-23T11:57:22Z", "parentGroupId": "", "path": null, "parentPath": "", "organisationUnit": false
      }, "role": "AUTHORIZED_CLERK"
    }
  ], "keystoreAlias": null, "isEnabled": true, "isAdministrator": false, "origin": "internal", "enabled": true, "administrator": false
}
```

If the ST has not been successfully validated the HTTP response code is 401 Unauthenticated and it contains a JSON object with the error description:

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: PUT, POST, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: content-type, x-requested-with, accept, request-id, origin, location, authorization
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=ISO-8859-1
Content-Length: 137
Date: Wed, 13 Dec 2017 13:36:08 GMT
```

```
{
  "cause": "Ticket 'ST-6-ctJno0PYDtybWNcBG5qU-CSD-04' not recognized",
  "message": "Ticket 'ST-6-ctJno0PYDtybWNcBG5qU-CSD-04' not recognized"
}
```

3.3.6 Validating a Service Ticket

The validation of the ST is built inside the RINA CPI, which automatically calls the REST API of the CAS server to validate the service ticket. However, this validation can be also triggered explicitly (in this case you cannot log in with the same ST also to RINA because the CAS protocol specifies, that each service ticket is valid only once. When using RINA CPI, this kind of validation must not be used and is mentioned here only because of testing purposes):

```
Request URL: http://server.example.com:8080/eessiCas/p3/serviceValidate?service=http://server.example.com:8080/eessiCpi&ticket=ST-7-TRaZGtOgg1xrAaYkqaZB-CSD-04
Request Method: GET
```

An example response of successful validation contains the information about the authentication possibly with some user attributes:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
```

```
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Type: text/html; charset=UTF-8
Content-Language: en-GB
Transfer-Encoding: chunked
Date: Wed, 13 Dec 2017 13:46:47 GMT

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationSuccess>
    <cas:user>supervisor</cas:user>
    <cas:attributes>

<cas:longTermAuthenticationRequestTokenUsed>false</cas:longTermAuthenticationRequestTokenUsed>
    <cas:isFromNewLogin>false</cas:isFromNewLogin>
    <cas:authenticationDate>2017-12-
13T14:40:35.327+01:00[Europe/Berlin]</cas:authenticationDate>
    <cas:authenticationMethod>eessiUserAuthenticationHandler</cas:authenticationMethod>

<cas:successfulAuthenticationHandlers>eessiUserAuthenticationHandler</cas:successfulAuthenticatio
nHandlers>
    </cas:attributes>
  </cas:authenticationSuccess>
</cas:serviceResponse>
```

An example response of unsuccessful validation:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Type: text/html; charset=UTF-8
Content-Language: en-GB
Transfer-Encoding: chunked
Date: Wed, 13 Dec 2017 13:55:36 GMT

<cas:serviceResponse xmlns:cas='http://www.yale.edu/tp/cas'>
  <cas:authenticationFailure code="INVALID_TICKET">Ticket &#39;ST-12-LlAiaodH6Em24GipYKLq-CSD-
04&#39; not recognized</cas:authenticationFailure>
</cas:serviceResponse>
```

3.4 CSRF Protection

The CPI REST application utilizes the CSRF protection by implementing an XSRF token. This token is generated upon a successful login request to the CPI with the correct service ticket and the service ID parameter. The token is bound to the session and must be sent

with every subsequent request to the CPI. The token is sent in the response to the login call in a header named "X-Auth-Cookie" in a specific header named "X-XSRF-TOKEN".

Following snippet shows the response of the login CPI call:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: PUT, POST, GET, OPTIONS, DELETE
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: content-type, x-requested-with, accept, request-id, origin,
location, authorization
X-Auth-Cookie: 44b0137d-4fc0-49da-a26d-280dc1f8de9b
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: JSESSIONID=A1D904D6A6F7C416F37CDED4C1DC217E.server1; Path=/eessi-rina-cpi/; HttpOnly
Content-Length: 1013
Date: Wed, 13 Dec 2017 13:41:41 GMT

{"id":"1","version":12,"username":"supervisor","institutionId":"LOOPBACK","firstName":"John","last
tName":"Doe","creationDate":"2017-08-
23T11:57:22Z","password":null,"phoneNumber":null,"email":"supervisor@example.com","memberships":[
{"group":{"id":"1","version":9,"name":"ACME","institutionId":"LOOPBACK","isOrganisationUnit":fals
e,"needReassignment":false,"displayName":"ACME","description":"Test group of institution
LOOPBACK","creationDate":"2017-08-
23T11:57:22Z","parentGroupId":"","path":null,"parentPath":"","organisationUnit":false},"role":"SU
PERVISOR"}, {"group":{"id":"1","version":9,"name":"ACME","institutionId":"LOOPBACK","isOrganisatio
nUnit":false,"needReassignment":false,"displayName":"ACME","description":"Test group of
institution LOOPBACK","creationDate":"2017-08-
23T11:57:22Z","parentGroupId":"","path":null,"parentPath":"","organisationUnit":false},"role":"AU
THORIZED_CLERK"}], "keystoreAlias":null, "isEnabled":true, "isAdministrator":false, "origin":"interna
l", "enabled":true, "administrator":false}
```

The token must be then stored and added to all the nexts requests in the "X-XSRF-TOKEN" header:

```
GET http://localhost:8080/eessi-rina-cpi/ApplicationProfile
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Cookie: JSESSIONID=96B19F516F7B79F90F3BAC69225026DA.server1;
Host: localhost:8080
Origin: http://localhost
Referer: http://localhost/
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.102 Safari/537.36
X-XSRF-TOKEN: 44b0137d-4fc0-49da-a26d-280dc1f8de9b
```

3.5 CORS Filtering

To enable deployment of the RINA modules in different domains, the RINA CPI implements a CORS filter, which can be configured in two ways:

- Allowed Origins is set to wildcard "*" – in this case all the origins will be allowed to access CPI. This configuration is default.
- Restrict the allowed origins to a predefined set of addresses

The configuration is takes place in the configuration files defined in the following JVM arguments of the REST CPI container:

- rina.configuration.cas
- rina.configuration.casreports

The configuration files defined in the above mentioned JVM arguments can specify the following parameters:

```
authentication.cors.allowedOriginAll=true  
authentication.cors.allowedOrigins=http://subdo1.localhost,http://subdo1.localhost
```

If the parameter "*authentication.cors.allowedOriginAll*" is set to **true**, clients from all origins will be allowed, hence the second argument will be ignored.

If the parameter "*authentication.cors.allowedOriginAll*" is set to **false**, only clients from from the origins specified in "*authentication.cors.allowedOrigins*" will be granted access in the context of the CORS settings. This parameter is a comma-separated list of values.

3.6 Authorisation to access protected resources

The client accesses protected resources by presenting the JSESSIONID cookie to the resource server.

The resource server creates the session context based on this cookie and it contains the granted authorities and possibly other information needed to perform access rights validation for the resources.

RINA consists of two consoles with different purposes:

- Administrative console and
- Case management console.

Each console accesses a different type of REST resources and the user types are defined accordingly (some of the resources may be shared, however, may differe in data contained):

- Administrator - has the full access to the administrative REST resources.
Administrator user is common for all Tenants.
- Regular users (clerk) - the access of the clerk role to case management REST resources is explicitly defined per case (see case assignment).

The regular users can have more roles within the organisation such as:

- Manager / Supervisor
- Authorised clerk
- Unauthorised clerk
- Auditor
- Medical user
- VIP user

It is important to note, that these roles are not global and are always assigned to a single membership in a group. I.e. a user can be assigned a role SUPERVISOR in the GROUP1 but he might have a role AUDITOR in the GROUP2.

These memberships (together with assignment policies) define if a user is allowed to create a case (CREATOR policy) and what operations is he allowed to perform (CASE policy).

As shown in the table below, depending on the user role (also known as actor), there are different pre-defined actions, which the respective user can perform on a case.

Additionally, a user can have more than one role inside the organization. More than this, there is hierarchical groups and users organisational structure and a user can have different roles in different groups.

User access privileges can be associated with business use cases and therefore this will be provided as implicit privileges to the users who are assigned on respective cases.

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	✓			✓				

* 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

Important 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;

*Requests from Users, who do not have Medical Role assigned, to retrieve Medical attachment metadata, will return, instead of the real value of the name and filename, the following text: **"Name withheld for confidentiality reasons"**.*

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

Supervisors, Authorised clerks and non-authorised clerks 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.

3.7 Data Validation

This interface can be configured to perform validation on all the SED documents that are submitted to the system.

There are two types of data validation

- Data validation at SED level (data entry validations)
- Data validation at message level

3.7.1 Data validation at SED level

The data validation at the SED level consist in the following:

- Validations made at user interface (UI) level
- Validations made at the server level

Validations made at user interface (UI) level

Validations made at user interface (UI) level are done in the client tier, in the browser

Field used: `ApplicationProfileDto.validateSED`

Validations made at the server level

Validations are made in the server tier, during the saving process.

Field used: `ApplicationProfileDto.sedValidationMode`

SED validation in this layer can be enabled with the property ***sedValidationMode*** from the object ***ApplicationProfileDto*** that is used to configure the system.

When enabled, there are 3 working validation modes:

- NO VALIDATION – *value used to disable the validation altogether*
- CONTINUE WHEN NO VALIDATION – Ignore validation and move on even if no XSDs are to be found for validating the SED XML. *If the XSD is found then the validation is performed.* EXCEPTION WHEN NO VALIDATION – Stop *by throwing an error if no XSDs are to be found for validating the SED XML. If the XSD is found then the validation is performed.* Each and every time a document is submitted the JSON payload that represents the SED is going to be transformed on the fly into an XML SED that is validated against its counterparty SED XSD.

The XSD path is configured with the property ***xsdRepositoryPath*** from the

MessagingSettingsDto and must have the following directory structure:

- sbdh – This folder contains the standard business document header XSDs
- sed – This folder contains all the SED XSDs
- transaction – This folder contains the business messaging protocol XSDs

3.7.1.1 Exceptions

For the validations made at SED level there are two types of exceptions which consist in:

- Primary errors – generated at save time (or when saving the SED) by validating against the DRAFT XSD

- Extended errors (warnings) – generated at send time (or when sending the SED) by validating against the FULL XSD

3.7.2 Data validation at message level

The data validation at message level can be configured by setting up the following field: `ApplicationProfileDto.bmpValidationMode`.

SED validation in this layer can be enabled with the property ***bmpValidationMode*** from the object ***ApplicationProfileDto*** that is used to configure the system.

When enabled, there are 3 working validation modes:

- NO VALIDATION – value used to disable the validation altogether
- CONTINUE WHEN NO VALIDATION – Ignore validation and move on if no XSDs are to be found for validating the SED XML. If the XSD is found then the validation is performed.
- EXCEPTION WHEN NO VALIDATION – Stop by throwing an error if no XSDs are to be found for validating the SED XML. If the XSD is found then the validation is performed.

3.8 Audit trail

The audit trail is a sequence of chronological records of EESSI RINA system or user related events to enable the investigation of different type of incidents. These (business) events belong to the case management, message exchange and the application (i.e. configuration elements) context. The information in audit trail assists investigations of suspected incidents and efficient research in terms of user accountability and reconstruction of events.

The audit trail is a fundamental part of EESSI system and this section aims to provide the reader with an introduction on the rules applied in RINA to capture and handle audit trail data, the audit trail architecture and its components, as well as an overview on the data model.

The complete information about Audit Trail architecture can be found in the *RINA Architecture Overview*² document.

3.8.1 RINA Audit Trail Architecture

In the multi layered RINA, CPS and BMS components are participating in the overall audit trail architecture, as they are involved in the most business significant operations. Both are sharing a common audit model, being responsible for different areas and type of events.

The diagram below provides the audit trail mechanism of those components.

² Available in, together with other document, in the Confluence space for RINA Architecture: <https://citnet.tech.ec.europa.eu/CITnet/confluence/display/EESSI/RINA++Architecture+Documentation>

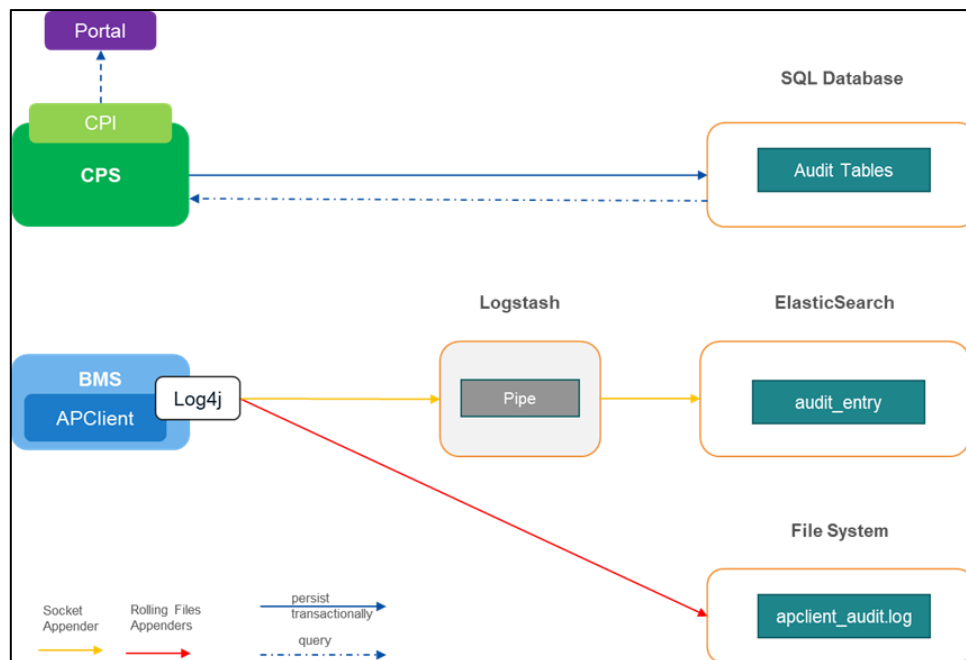


Figure 3: RINA CPS Audit Trail architecture – Component view

As explained in the following sections, the content of the BMS layer also becomes available through the CPS layer.

3.8.2 Audit Trail Persistence & Retrieval

Case Processing Services, as the component responsible for the most user business operations, produce and store the audit logs (events) into the PostgreSQL Database.

The generation of the audit records is performed as part of the same business transaction with the triggering operation. This ensures that every auditable operation will always be audited either this operation has a successful outcome or not.

The audit events of CPS can be retrieved by CPI or Portal. Admin users have access to all the events produced by CPS whereas normal business users have access to events that are related to the resources they have sufficient audit access (cases, documents etc).

In addition, for both repositories, it is possible to plug a custom National Application audit aggregation service into the PostgreSQL of CPS (or/and Elasticsearch)

3.8.3 General approach to audit trail data model

All different RINA components use the same Audit Trail Data model to enable consistency and easier aggregation.

The following diagram shows the set of UML classes involved in RINA Audit Trail, as well as their relationships. Right after it, a detailed description of each class as well as its attributes is provided to the reader.

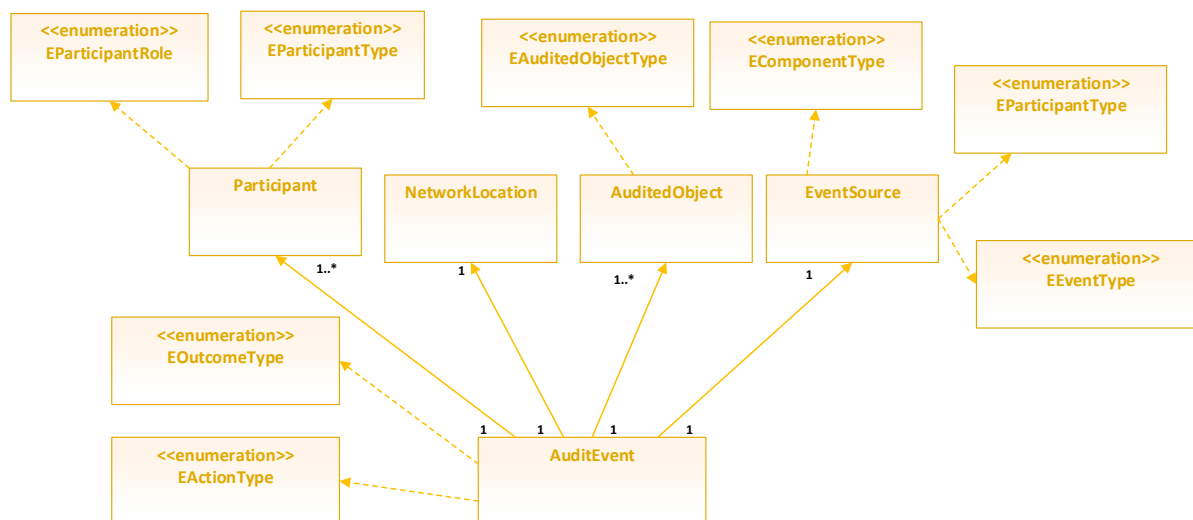


Figure 4: RINA Audit Data Model

Regarding the different domains of the Audit Model, there are two different representations

- SQL Tables for CPS in PostreSQL.
- JSON Document for CPI (CPS component layer).

An audit event is any triggered event for which a record will be created in the audit-trailing repository.

An audit event is identified by its ID and contains all the data shown in the next table.

Attribute	Description	Data Type
Id	The ID of the audit event	String
Action	The type of action which is executed which can be one of the following: create, read, update, delete, execute	EActionType
userName	The username of the user who triggered the action	String
networkLocation	The network location are the identification details (machine & IP) on which the action has been triggered	NetworkLocation
Date	Date of the audit event	DateTime
source	The source of the event which has three attributes (category, component, type)	EventSource

outcome	The outcome of the action triggered which can be success, error or unauthorised	EOutcomeType
outcomeDetails	When outcome is error or unauthorised, the outcomeError contains the error message.	String
auditedObjects	The audited objects which are identified by id, type and details	List<AuditedObject>
participants	The list of participants to the triggered action.	List<Participant>
tenantId*	The tenantId of the tenant that triggered the action.	String

* *tenantId* is only available in the persisted representation (under SQL for CPS component layer, under File/ES for BMS component layer) – hence not available through the CPI or Admin Portal.

3.8.3.1 Audited Object

Defines the details about the audit object and important/relevant details of the respective audit object which need to be recorded in the audit trail about respective event.

Attribute	Description	Data Type
Id	The id of the audited object	String
Type	The type of the audited object (e.g. attachment, case credential, policy, document, etc.)	EAuditedObjectType
details	In case of error the object triggered by REST shall be recorded here	String

3.8.3.2 Event Source

Details and identify the source of an event.

Attribute	Description	Data Type
categoryType	The category of an event source (i.e. messaging, security, business)	ECategoryType
componentType	The component of an event source (e.g. attachment, case, business messaging, technical messaging, etc.)	EComponentType
eventType	The type of an event source which consist in the actual action which triggered the event	EEventType

3.8.3.3 Network Location

This defines the network location details for a triggered event. These details will consist in the machine on which the event was triggered and its corresponding IP in the network.

Attribute	Description	Data Type
machine	The machine from which the action has been triggered	String
Ip	The IP of the machine	String

3.8.3.4 Participant

This describes the participant(s) which are taking part to a triggered event.

Attribute	Description	Data Type
Id	The id of the participant	String
Type	The type of the participant which can be person or organisation	EParticipantType
Role	The role of participant in a triggered event/audited object (i.e. sender, receiver, subject)	EParticipantRole

3.8.3.5 EActionType enumeration values and description

The action type describes the actions which correspond to a triggered event. This can be one of the following: create, read, update, delete, execute.

Value	Name	Description
Cre	CREATE	Creation of an audited object type
Re	READ	Read action of an audited object type
upd	UPDATE	Update an audited object type
del	DELTE	Delete an audited object type
exec	EXECUTE	Execution action (when none of the above are happen)

Following values are supported for the action type and can be used to filter the messages by the CPI Endpoint or Portal:

- Read
- Create
- Update
- Delete
- Execute

3.8.3.6 EAuditedObjectType enumeration

The audited object type defines the relevant audited objects for a triggered event.

In case of a triggered event which involves actions on more types for each audited object type an audited object will be appended to the audited object list(e.g. for an event which will consist in submitting and attachment to a case, there will be two audited objects: one for the attachment and one for the case).

Please note that even though the types for audited objects are similar with the component types these are intended for a different use.

Value	Name	Description
attch	ATTACHMENT	Used when the triggered event involves an action pertaining to an attachment (i.e. submit/ delete/ retrieve an attachment to a case or document)
cas	CASE	Used when the triggered event involves also an action pertaining to a case
cmt	COMMENT	Used when the triggered event involves an action pertaining to a comment (i.e. submit or delete a comment to/from a case or document)
alarm	ALARM	Used when the triggered event involves an action regarding an alarm (e.g. setting or clearing and alarm)
doc	DOCUMENT	Used when the triggered event involves an action pertaining to a document/SED (e.g. actions like submit/ retrieve documents, actions like submitting/ deleting/ etc. attachments or comments to a document, etc.)
sdoc	SUBDOCUMENT	Used when the triggered event involves an action pertaining to a subdocument (e.g. actions like submit/ retrieve subdocuments, actions like submitting/ deleting/ etc...)
notf	NOTIFICATION	Used when the triggered event involves an action pertaining to a notification (i.e. retrieve notification details/ summary of notifications)
srch	SEARCHDEFINITION	Used when the triggered event involves an action pertaining to a search definition (i.e. create, update, delete, or use of a search definition when search case by search definition)
usgr	USER_GROUP	Used when the triggered event involves an action pertaining to a user or group (i.e. user/group creation / deletion)
uspr	USER_PROFILE	Used when the triggered event involves an action pertaining to a user profile (i.e. user profile update)
appr	APPLICATION_PROFILE	Used when the triggered event involves an action pertaining to the application profile
cred	CREDENTIAL	Used when the triggered event involves a login or logout actions

Value	Name	Description
poly	POLICY	Used when the triggered event involves an action pertaining to an authorisation policy (e.g. actor assignment)
bmsg	BUSINESS_MESSAGE	Used when the triggered event involves an action pertaining to a business message
tmsg	TECHNICAL_MESSAGE	Used when the triggered event involves an action pertaining to a technical message (ebMS 3 received or sent user message)
ackm	BUSINESS_ACK	Used when the triggered event involves an action pertaining to an acknowledgement technical message (ebMS3 received or sent signal acknowledgement message)
errm	BUSINESS_ERROR	Used when the triggered event involves an action pertaining to an error technical message (ebMS3 received or sent signal error message)

Following values are supported for the object type and can be used to filter the messages by the CPI Endpoint or Portal:

- Notification
- Notification Type
- Credential
- Business Error
- Business Acknowledgement
- Attachment
- Comment
- Alarm
- Subdocument
- Business Message
- Search Definition
- Case
- User Profile
- Policy
- Document
- Technical Message
- Application Profile
- User Group

3.8.3.7 ECategoryType enumeration values and description

The category type is used in order to specify the category of an event source. This can be one of the following: messaging, security or business and are corresponding to RINA layers. The presentation layer is out of scope for the audit trail.

Attribute	Name	Description
mess	MESSAGING	Used for event source when the triggered action involves messaging events/messaging layer
sec	SECURITY	Used for event source when the triggered action involves security events (e.g. actions impacting the users, group, credentials, policies etc.)
bus	BUSINESS	Used for event source when the triggered action involves business events/business layer

Following values are supported for the category types and can be used for filtering messages by the CPI Endpoint or Portal:

- Security
- Business
- Messaging

3.8.3.8 Component Type enumeration values and description

Defines the component types relevant for an event source.

Value	Name	Description
Attch	ATTACHMENTS	Will be chosen if the triggered event involves the attachments component
Cas	CASES	Will be chosen if the triggered event involves the cases component
cmt	COMMENTS	Will be chosen if the triggered event involves the comments component
doc	DOCUMENTS	Will be chosen if the triggered event involves the documents component
notf	NOTIFICATIONS	Will be chosen if the triggered event involves the notifications component
srch	SEARCH_DEFINITIONS	Will be chosen if the triggered event involves the search definition component
Sec	SECURITY	Will be chosen if the triggered event involves the security component
admin	ADMINISTRATION	Will be chosen if the triggered event involves the administration component

bmsg	BUSINESS_MESSAGING	Will be chosen if the triggered event involves the business messaging component
tmsg	TECHNICAL_MESSAGING	Will be chosen if the triggered event involves the technical messaging component

Following values are supported for the component types and can be used for filtering messages by the CPI Endpoint or Portal:

- Notifications
- Security
- Search Definitions
- Business Messaging
- Cases
- Documents
- Administration
- Attachments
- Comments
- Technical Messaging

3.8.3.9 EEventType values and description

Defines all types of the events, which trigger the audit logging process.

Value	Name	Description
sub_attch_on_cas	SUBMIT_ATTACHMENT_ON_CASE	An event generated during submit attachment on a case
del_attch_frm_cas	DELETE_ATTACHMENT_ON_CASE	An event generated during delete attachment of a case
retr_attch_of_cas	RETRIEVE_ATTACHMENT_ON_CASE	An event generated during retrieve attachment of a case
sub_attch_on_doc	SUBMIT_ATTACHMENT_ON_DOCUMENT	An event generated during submit attachment on a document
retr_attch_of_doc	RETRIEVE_ATTACHMENT_ON_DOCUMENT	An event generated during retrieve attachment of a document
del_attch_of_doc	DELETE_ATTACHMENT_ON_DOCUMENT	An event generated during delete attachment of a document
srch_cas_by_def_and_or_txt	SEARCH_CASES_BY_SEARCH_DEFINITION_AND_OR_FREE_TEXT	An event generated during search cases based on definition and/or free text

Value	Name	Description
cre_new_cas	CREATE_NEW_CASE	An event generated during create new case
retr_cas_by_id	RETRIEVE_CASE_BY_ID	An event generated during get a case by its local Case id
retr_cas_by_international_id	RETRIEVE_CASE_BY_INTERNATIONAL_ID	An event generated during get the details of a case by its international id
retr_cas_id_by_international_id	RETRIEVE_CASE_ID_BY_INTERNATIONAL_ID	An event generated during get a local case id by its international id
set_alarm	SET_ALARM	An event generated when a new alarm is created
clr_alarm	CLEAR_ALARM	An event is generated when an alarm is cleared
retr_cas_by_business_id	RETRIEVE_CASE_BY_BUSINESS_ID	An event generated during get the details of a case or the local Case Id by its business id
assig_cas	ASSIGN_CASE	An event generated during assign a case
ar_un_cas	ARCHIVE_UNARCHIVE_CASE	An event generated when archive or unarchive
upd_cas_sensitive	UPDATE_CASE_SENSITIVE	An event generated when the 'sensitive' flag in the case metadata has been updated.
retr_cas_hash_by_id	RETRIEVE_CASE_HASHCODE_BY_ID	An event generated during get the case hash code of the case by the case id
imp_sdoc	IMPORT_BATCH	An event generated during import of the subdocuments
exp_sdoc	EXPORT_BATCH	An event generated during exporting the subdocuments
retr_cas_assig	RETRIEVE_CASE_ASSIGNMENTS	An event generated during get case assignment
sub_new_cmt_on_cas	SUBMIT_COMMENT_ON_CASE	An event generated during submit a new comment on a case
del_cmt_of_cas	DELETE_COMMENT_ON_CASE	An event generated during delete a comment of a case

Value	Name	Description
sub_new_cmt_on_doc	SUBMIT_COMMENT_ON_DOCUMENT	An event generated during submit a new comment on a document
del_cmt_of_doc	DELETE_COMMENT_ON_DOCUMENT	An event generated during delete a comment of a document
retr_init_doc_of_cas	RETRIEVE_INITIAL_DOCUMENT	An event generated during retrieve initial Document of an action. An event when retrieving the Initial Document Content or the Admin Document Content. That covers admin case operations i.e.: <i>Close, Reopen, Delete case, Local Close, Local Reopen, Send Participants, Select Participants, Read Participants, Update Participants, Request Approval, Remove Subdocument</i>
cre_doc	CREATE_DOCUMENT	An event generated when submitting the Document first time
upd_doc	UPDATE_DOCUMENT	An event generated when submitting the Document subsequently
del_doc	DELETE_DOCUMENT	An event generated when deleting the Document
snd_doc	SEND_DOCUMENT	An event generated when sending the Document to counterparties
sub_doc	SUBMIT_DOCUMENT	An event when submitting the Document Content or the Admin Document Content. That covers admin case operations i.e. <i>Close, Reopen, Delete case, Local Close, Local Reopen, Send Participants, Select Participants, Read Participants, Update Participants, Request Approval, Remove Subdocument</i>
retr_doc	RETRIEVE_DOCUMENT	An event generated during retrieve Document
retr_thumb	RETRIEVE_THUMBNAIL	An event generated during retrieve thumbnail

Value	Name	Description
login	LOG_IN	An event generated during login
logout	LOG_OUT	An event generated during logout
retr_notf_details	RETRIEVE_NOTIFICATIONS_DETAILS	An event generated during retrieve the notifications details
retr_cnsld_summ_notf_usr	RETRIEVE_NOTIFICATIONS_CONSOLIDATED_SUMMARY	An event generated during retrieve the consolidated summary of notifications for a user (errors, warnings, information)
retr_summ_notf_usr	RETRIEVE_NOTIFICATION_SUMMARY	An event generated during retrieve the summary of notifications for a user
retr_tsno_and_notf_on_ts	RETRIEVE_NOTIFICATION_TIME_SLOTS	An event generated during retrieve the time slots and the number of notifications on each time slot
upd_notf	UPDATE_NOTIFICATION	An event generated during update a notification
cre_new_srch_def	CREATE_SEARCH_DEFINITION	An event generated during create a new Search Definition
upd_spec_srch_def	UPDATE_SEARCH_DEFINITION	An event generated during update a specific Search Definition
del_spec_srch_def	DELETE_SEARCH_DEFINITION	An event generated during delete a specific Search Definition
upd_usr_prof	UPDATE_USER_PROFILE	An event generated during update the User Profile
upd_app_prof	UPDATE_APPLICATION_PROFILE	An event generated during update the Application Profile
usr_gr_cre	CREATE_USER_GROUP	An event generated during user/group creation
usr_gr_del	DELETE_USER_GROUP	An event generated during user/group deletion
auth_pol_chg	CHANGE_AUTHORIZATION_POLICY	An event generated during authorisation policy change
send_bmsg	SEND_BUSINESS_MESSAGE	An event generated during send a business messages to a preconfigured EESSI access point using the Holodeck AS4
notif_reg_lsn_bmsg	NOTIFY_ABOUT_RECEIVED_BUSINESS_MESSAGE	An event generated during notify a register listener about a received business message

Value	Name	Description
notif_reg_lsn_stat_bmsg	NOTIFY_ABOUT_STATUS_UPDATE	An event generated during notify a register listener about a status update of a business message (ack or error)
send_tmsg	SEND_TECHNICAL_MESSAGE	An event generated during send a technical message to a preconfigured EESSI access point using the Holodec AS4 (user message, pull request, ack., error)
rec_tmsg	RECEIVE_TECHNICAL_MESSAGE	An event generated during receive a technical message from a preconfigured EESSI access Point using the Holodeck AS4 (user message, Ack., error)
app_start	APPLICATION_START	An event generated during the start of the application
app_stop	APPLICATION_END	An event generated during the stop of the application
rec_new_cas	RECEIVE_NEW_CASE	An event generated when a new Case is received
rec_bmsg	RECEIVE_BUSINESS_MESSAGE	An event is generated when new Business Message is received on CPS component layer
pr_rec_bmsg	PROCESS_RECEIVED_MESSAGE	An event is generated when a received Business Message is processed on CPS component layer

Following values are supported for the event types and can be used for filtering messages, by the CPI Endpoint or Portal:

- Submit Attachment On Document
- Create Document
- Import Subdocument Batch
- Create Search Definition
- Logout
- Delete Attachment On Document
- Export Subdocument Batch
- Application Start
- Submit Document
- Receive Technical Message
- Change Authorization Policy
- Retrieve Notifications Consolidated Summary
- Update Document
- Send Document

- Update Search Definition
- Retrieve Initial Document
- Set Alarm
- Delete Attachment on Case
- Retrieve Case By Id
- Create User Or Group
- Retrieve Attachment On Document
- Submit Attachment on Case
- Retrieve Case Hash Code By Id
- Delete Comment On Document
- Retrieve Notification Time Slots
- Retrieve Case Assignments
- Login
- Submit Comment On Document
- Update User Profile
- Retrieve Attachment on Case
- Clear Alarm
- Update Application Profile
- Delete Document
- Retrieve Notification Summary
- Retrieve Document
- Delete Comment On Case
- Notify About Received Business Message
- Submit Comment On Case
- Assign Case
- Send Technical Message
- Retrieve Notification Details
- Delete User Or Group
- Notify About Status Update
- Create New Case
- Retrieve Thumbnail
- Application End
- Delete Search Definition
- Search Cases By Search Definition And Or FreeText
- Update Notification
- Receive New Case
- Receive Business Message (only CPI)
- Process Received Message (only CPI)

3.8.3.10 EOutcomeType enumeration values and description

The outcome type is an audit event attributes which specify how a triggered action ended. This can be one of the following: success, error, or unauthorised.

Value	Name	Description
succ	SUCCESS	Used when the action which generates the audit event ends with success.
err	ERROR	Used mainly when HTTP 500 type of exception are occurring
rjct	UNAUTHORISED	Used mainly when HTTP 401 & HTTP 403 types of exceptions are occurring

3.8.3.11 EParticipantRole enumeration values and description

This section describes the participant role in an audited event.

Value	Name	Description
sndr	SENDER	Used if the event is in context of messaging and the participant is the sender institution
rcvr	RECEIVER	Used if the event is in context of messaging and the participant is the receiver institution
subj	SUBJECT	Used in context of non-messaging events

Following values are supported for the participant role and can be used for filtering the messages:

- Receiver
- Sender
- Subject

3.8.3.12 EParticipantType enumeration values and description

This section describes the participant type in an audited event, which can be person or organisation depending on the event specifics.

Value	Name	Description
per	PERSON	Used if the audited event participant is a person
Org	ORGANISATION	Used if the audited event participant is non-person.

Following values are supported for the participant type and can be used for filtering the messages:

- Organization
- Person

3.8.4 Particular approach of audit trailing for Case Processing Interface

This mapping aim to illustrate what are the possible associations between all the categories, components, events and their corresponding audited objects (details).

Category	Component	Event	Audited Objects
Business	#ATTACHMENTS: operations on Attachments	SUBMIT_ATTACHMENT_ON_CASE	➤ Attachment ➤ Case
		DELETE_ATTACHMENT_ON_CASE	➤ Attachment ➤ Case
		RETRIEVE_ATTACHMENT_ON_CASE	➤ Attachment ➤ Case
		SUBMIT_ATTACHMENT_ON_DOCUMENT	➤ Attachment ➤ Case ➤ Document
		RETRIEVE_ATTACHMENT_ON_DOCUMENT	➤ Attachment ➤ Case ➤ Document
		DELETE_ATTACHMENT_ON_DOCUMENT	➤ Attachment ➤ Case ➤ Document
Business	#CASES: operations on Cases	SEARCH_CASES_BY_SEARCH_DEFINITION_AND_OR_FREE_TEXT	➤ Search text ➤ SearchDefinition Id (Optional)
		CREATE_NEW_CASE	➤ Case
		RECEIVE_NEW_CASE	➤ Case
		RETRIEVE_CASE_BY_ID	➤ Case
		UPDATE_CASE_SENSITIVE	➤ Case
		ASSIGN_CASE	➤ Case
		RETRIEVE_CASE_HASHCODE_BY_ID	➤ Case
		RETRIEVE_CASE_ASSIGNMENTS	➤ Case
Business	#COMMENTS: operations on Comments	SUBMIT_COMMENT_ON_CASE	➤ Case ➤ Comment
		DELETE_COMMENT_ON_CASE	➤ Case ➤ Comment
		SUBMIT_COMMENT_ON_DOCUMENT	➤ Case ➤ Document ➤ Comment
		DELETE_COMMENT_ON_DOCUMENT	➤ Case ➤ Document ➤ Comment

Category	Component	Event	Audited Objects
Business	#DOCUMENTS (incl. thumbnail): operations on Documents	RETRIEVE_INITIAL_DOCUMENT	➤ Case ➤ Action (id and name) ➤ Document (optional)
		SUBMIT_DOCUMENT	➤ Case ➤ Action (id and name) ➤ Document
		RETRIEVE_DOCUMENT	➤ Case ➤ Document
		RETRIEVE_THUMBNAIL	➤ Case ➤ Document
Business	# MESSAGING Operations on Messages	RECEIVE_BUSINESS_MESSAGE	➤ Case (international Id) ➤ Documentation
		PROCESS_RECEIVED_BUSINESS_MESSAGE	➤ Case (international Id) ➤ Documentation
		SEND_TECHNICAL_MESSAGE	➤ BusinessMessage ➤ Case ➤ Document
		NOTIFY_ABOUT_RECEIVED_BUSINESS_MESSAGE	➤ BusinessMessage ➤ Case ➤ Document
		NOTIFY_ABOUT_STATUS_UPDATE	➤ BusinessMessage ➤ Case ➤ Document
Security	#SECURITY: operations on login	LOG_IN	➤ Credential
		LOG_OUT	➤ Credential
		CREATE_USER_GROUP	➤ Credential
		DELETE_USER_GROUP	➤ Credential
		UPDATE_USER_GROUP	➤ Credential
		CHANGE_AUTHORIZATION_POLICY	➤ Policy
Business	#NOTIFICATION S: operations on Notifications	RETRIEVE_NOTIFICATIONS_DETAILS	➤ Notification (Ids) ➤ Case (Optional)

Category	Component	Event	Audited Objects
		RETRIEVE_NOTIFICATIONS_CONSOLIDATED_SUMMARY	➤ N/A
		RETRIEVE_NOTIFICATION_SUMMARY	➤ N/A
		RETRIEVE_NOTIFICATION_TIME_SLOTS	➤ N/A
		UPDATE_NOTIFICATION	➤ Notification
Business	#SEARCHDEFINITIONS: operations on SearchDefinitions	CREATE_SEARCH_DEFINITION	➤ SearchDefinition
		UPDATE_SEARCH_DEFINITION	➤ SearchDefinition
		DELETE_SEARCH_DEFINITION	➤ SearchDefinition
Business	# admin: operations on USER_PROFILE and APPLICATION_PROFILE	UPDATE_USER_PROFILE	➤ UserProfile
		UPDATE_APPLICATION_PROFILE	➤ ApplicationProfile
Security	#Admin	APPLICATION_START	➤ ApplicationProfile
		APPLICATION_STOP	➤ ApplicationProfile

3.9 HTTP Error codes

When an API request fails, the EESSI Case Processing Interface API returns an HTTP 4xx or 5xx status code that generically identifies the failure along with a JSON response that provides more specific information about the error that caused the failure.

All POST or PUT calls should always have a non-empty and valid json body. If the json is not present, then all POST and PUT calls will return a 500 response code and the reason will be "Required request body content is missing". If the json is not valid then all POST and PUT calls will return a 500 response code and the reason will be "Could not read JSON: Unrecognized token"

All PUT calls like "PUT UserProfile/ProcessDefinitionFields/{processDefinitionName}" should always have a non-empty *processDefinitionName*. If the id is not present, then all PUT calls will return a 404 error code and an html error description: "The requested resource is not available".

All GET calls like GET /Cases/{caseId}/Documents/{id} should have non empty parameters, otherwise the response will be 404 error code and an html error description: "The requested resource is not available".

For each error, the JSON response will include a **reason** field and optionally a **stack** field.

Note: The content of the **stack** field is for debugging purpose only and should not be displayed to the end user. The client is responsible for displaying an appropriate localized error message to the end user based on the error code returned.

If an API request is successful, no error related fields should be expected to be present in the response.

Example response body on Error

```
{  
  "error": "Could not get case",  
  "error_description": "Case id '123' does not exist",  
  "stack": "unexpected error occurred",  
}
```

The following table shows the common errors that can be returned when an API call fails:

HTTP Status Code	Message	Description
401	Unauthorized	Authentication is required and has failed
403	Access denied	Access to the specified resource has been forbidden
404	Not Found	The requested resource could not be found but may be available again in the future
500	Internal Server Error	A generic error message, given when an unexpected condition was encountered and no more specific message is suitable

NOTE: standard HTTP codes 1xx, 2xx and 3xx are returned by the REST API, which are handled by client app

4 Functions and use cases

The Case Management Function (see Figure 5) provides capabilities to find cases based on sets of predefined criteria and to manage the messages that belong to a case based on system provided instructions.

This module addresses scenarios where the clerk is reacting to external events (e.g. direct relation with the citizens or institutions, system notifications on message exchange). This function allows the clerk to track all types of claims, streamline information entry with auto-population of forms based on the case data collected and tracked in the system, respond to regulatory and corporate body inquiries on the spot by having case details available. The Case Management, Validation and Guidance Module provide the necessary features to manage one or many active cases in the same time, in a structured manner.

The module is able to determine in a simplified manner what the status of the case is, which are the already exchanged documents, which are the next potential actions that can be executed and to visualize the case context specific data exchange guidance.

The module provides case specific notifications and alerts in a citizen/institution approach perspective and multiple views for visualizing a single case. In the following sections, the referenced core sub-system functions (Search Management, Case Management & Notification Management) are presented in parallel to the full list of the operations services as part of the RINA available interface (CPI) calls.

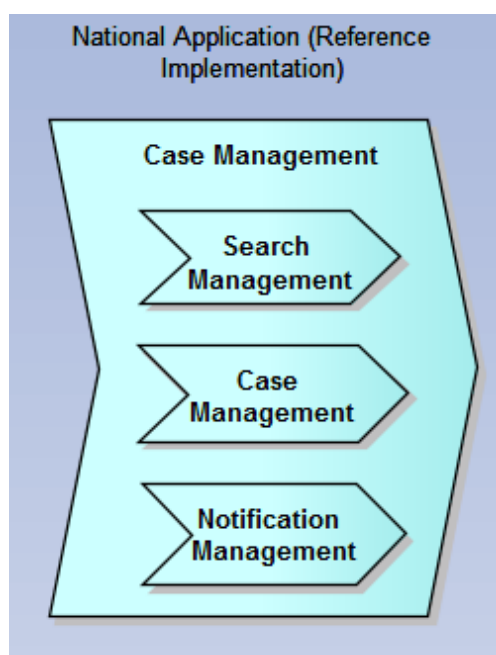


Figure 5. Categories of Case Management functions

The authorisation rules for all the methods within the management functions introduced below are detailed in the EESSI Rina CPI – Reference Documentation (under section 7) which is generated and provided as a separate document.

4.1 Search management function

The search management function provides tools for the clerk to be able to search and find cases in different ways: ad-hoc search (normal free-text search) or predefined-search.

The following figure (Figure 6) shows the use case diagram for this function:

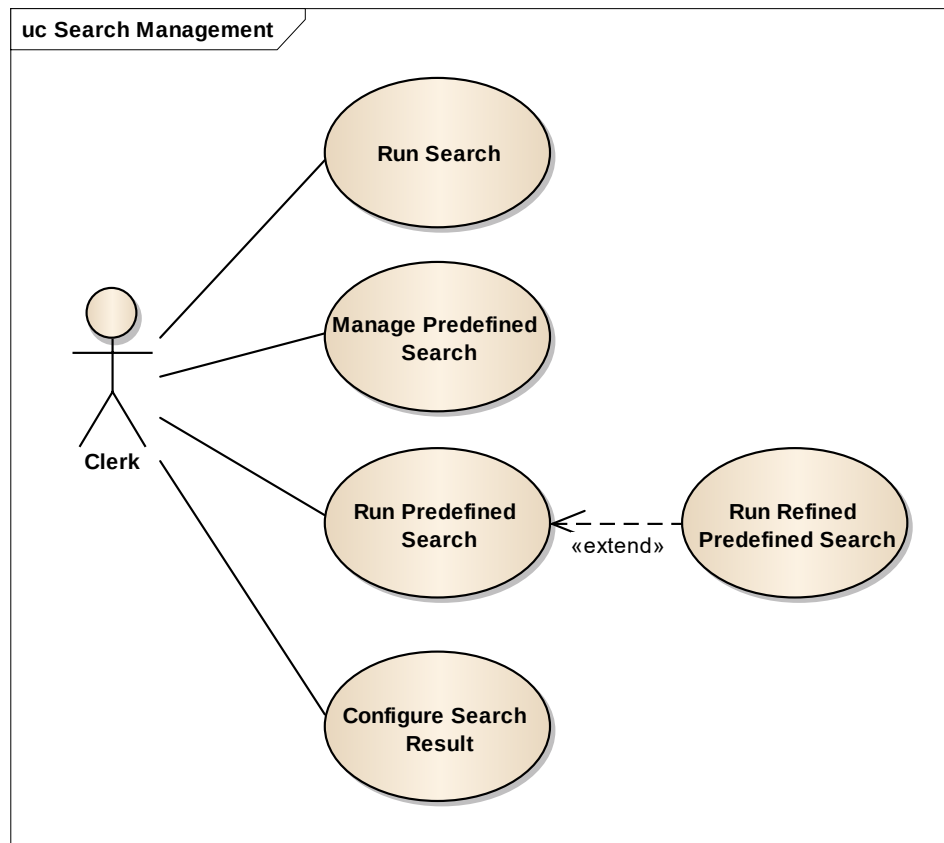


Figure 6. Search management function use case

4.1.1 Run search

Clerk uses ad-hoc search criterions:

- Clerk enters the search criterion (ex. partial string searching for the last name) in the form of text in the filter text-box and then clicks on filter button;
- If we are referring to partial search then after the partial search term entered the user needs to enter a star ('*') in order to signal to the application that this is partial search. Eg: searching for last name 'patt*' could return 'Patterson'.
- System identifies the case metadata that the query free-text parameters are referring to, applies the search parameters correspondingly and returns the list of cases that correspond to these criterions.

4.1.2 Manage predefined search

The system must be able to provide tools to the clerk for him to search and find cases in different ways: during his search for cases the clerk will be able to use ad-hoc search (typically a search field and a button "Filter" will be provided to the clerk) or predefined-search.

In order to minimize searching effort for the clerk (e.g.: my cases, unassigned cases, my old pension claims cases, etc.), the clerk is given the possibility to define predefined persistent searches. He should be able to add, modify or delete a predefined search, using visual search expressions. All the search expressions for a predefined search should have an associated colour and name.

The searching criteria are applied to the cases' metadata (e.g., case type and status, assigned users, case importance, case participants, etc.).

This use case includes the fact that the clerk creates/updates/deletes predefined search criterions, to be reused later for searching activities:

CREATION / UPDATE:

- The clerk enters the name of the predefined search;
- The clerk creates one or more search criterions (Case type, case status, case importance, assigned user, participant organization, etc.)
- The clerk saves the newly created / updated predefined search

DELETE:

- The clerk enters the name of the search;
- The clerk deletes the predefined search.

4.1.3 Run predefined search

The clerk should be able to define and to use predefined searches in order to minimize searching effort for the clerk (e.g.: my cases, unassigned cases, my old pension claims cases, etc.)

The clerk must be able to select one of the existing predefined searches.

In this use case the clerk selects a predefined search.

4.1.4 Run refined predefined search

After the system has filtered the cases on clerk's predefined search request, the clerk can read the list of filtered cases.

The clerk must be able to use the free text search to refine the returned results once these results have been filtered by the system. It is an optional additional filtering he can use.

This use case implements these presentation and refinement features.

All users are allowed to run refined predefined refined searches;

4.1.5 Configure Search Result

The clerk will get the possibility to configure the list of columns that will be returned by the search engine, also the order of the columns and the order of the results (ascending/descending sort).

This use case handles with these configurations as well.

4.1.6 TsVector implementation details

The last implementation of the RINA application the search is done in directly in Postgresql using a new feature called TsVector and TsQuery functionality. This is done by indexing all the search terms into lexemes (grammatical units) thus enabling the application search to behave very fast and more reliable than ever for complex terms taking account of plural/singular differences, abbreviations or different forms of the verbs at different conjugation times. For example if one of the words saved for search is 'dancing' and the user searches for 'dance', then the result would be successfully found.

The search implementation has been done in the same manner all across the application for the different functionalities in the RINA app where searching is relevant. Below you can find a detailed table of what search terms are available for every search functionality in the app:

No.	Search functionality	Available search terms(db columns)
1	Cases search	• Case sid • Surname • Name • Birthdate • Localpin • Status • Buc type • Buc name
2	Business Exceptions & Pending Messages Search	• Business exception sid • Pending messages sid • Business exception id • Reason • Pending message id • Sdbh • Content location • Action type • International case id • Is protected person • Should notify • Is processed • Problem • Cause
3	Institution (organisation) search	• Organisation sid • Organisation id • Organisation name • Acronym

		• Ap id • Ap name
4	Subdocument search	• Case sid • Subdocument sid • Subodcumet prefill data

4.2 Case management function

The case management function covers all aspects of case management, including creation of case, creation of document, creation of a counterparty list, sending documents, running business process specific actions, listing cases, documents or actions and configuration of case manager.

The following figure (Figure 7) shows the use case diagram for this function:

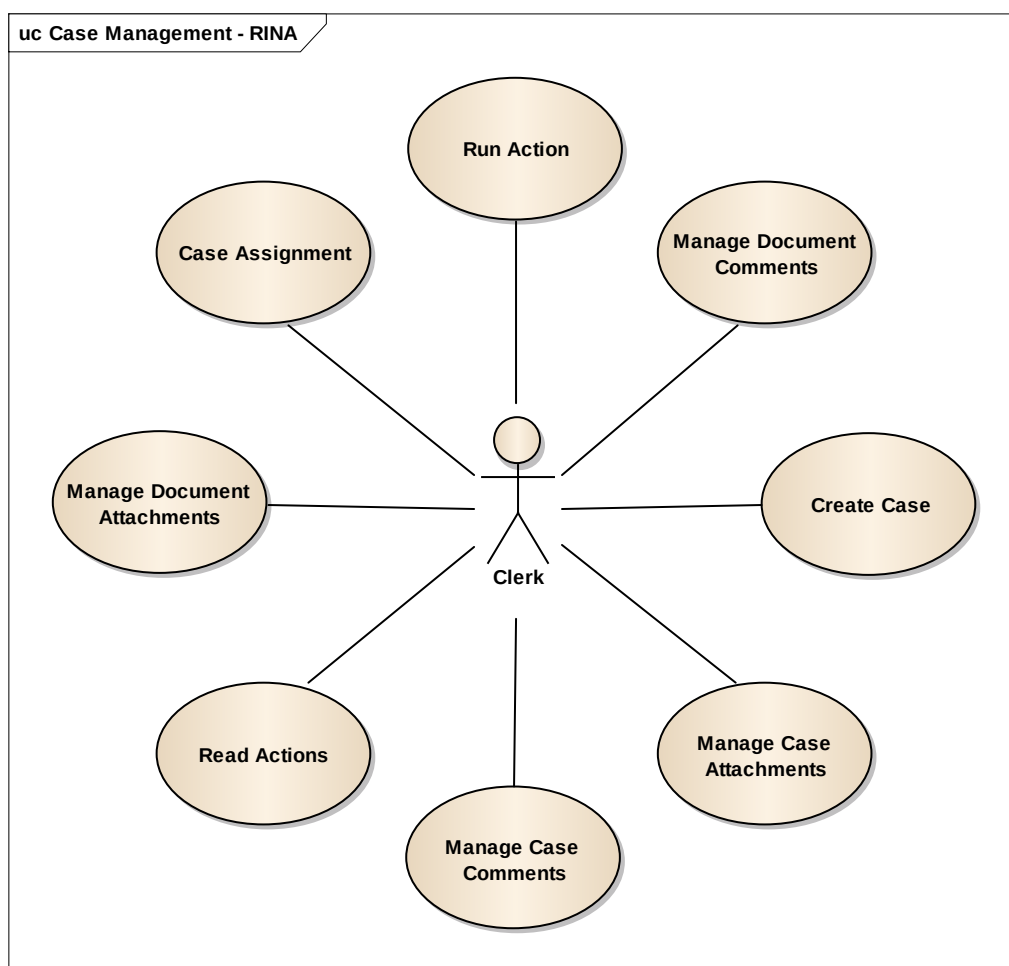


Figure 7. Case management function use case

4.2.1 Create case

The clerk creates a case explicitly:

- Clerk selects the subject if this exists (an organization or a citizen)

- Clerk accesses a hierarchical structure with the processes' definitions (type of cases). The default process definition proposed by the system to be instantiated is the last one used by the clerk and accessible for that type of subject.
- Clerk clicks on "New Case" link.

4.2.2 Read actions

The structured manner of case management is achieved using case type definition, and a list of actions related to a case must be displayed anytime depending on the status of the case: any operation executed by the clerk must be reflected in the case status. The available action list is recalculated based on the case type definition and the current process status. The relevant actions of a case are those actions which make sense in the current context.

Considering the complexity of the business use cases involved in EESSI, the list of actions returned must be structured, so that the user is able to easily determine:

- what the status of the case is;
- which are the already exchanged documents;
- which are the next potential actions to execute.

More specifically about SEDs, any SED has a specific behaviour. The SED's behaviour is mainly represented by the actions associated with it. These actions suggest the next step in the process (e.g. answer to a specific SED). The user must have the possibility to visualize and to execute all the document related actions, in association with the document preview.

The user must have the possibility to visualize all the available tasks / actions, grouped by subcategories or to visualize administrative or sector specific actions distinctively.

Possible classification:

- Sectorial actions not related to documents
- Administrative actions not related to documents
- Actions related to document instances (P2000, P6000...)

In fact, the Clerk needs to receive a refreshed available action list for the current active case.

4.2.3 Run action

The Clerk selects one action in the action list presented to him. The action list available at a given moment reflects the state of the case at that specific moment.

Note: *In order to execute a certain action, the action id need to be identified. The identification of the action can be performed exclusively through Get a case. The computation of next available actions is performed in a synchronoun way when we run/execute the action.*

Case level actions:

- Actions not related to documents (sectorial): ex. Create Case P_BUC_01
- Actions not related to documents (admin): ex. Reminder, Close Case, Reopen Case, Forward Case, Delete Case, Add Counterparties...

Document level actions:

- Actions related to document instances (P2000, P6000...) (not admin): ex. Create P2000, Edit Draft P2000, Send P2000, Update Sent P2000...
- Actions related to document instances (P2000, P6000...) (admin): ex. Invalidate P6000, Reject P6000...

Some possible actions are presented as "Extend" of this Run Action use case, because they present an important behaviour.

When update actions are involved, RINA stores / updates document data (SED content), version control and metadata (status, distribution list and any metadata defining the document stored) of the active document, and, consequently, updates the document repository.

When read actions are involved, RINA retrieves document data (SED content) and metadata (status, distribution list and any metadata defining the document stored) from storage location (document repository).

All document related actions are listed below:

- | | |
|---|---|
| ✓ Import Subdocuments | ✓ Retrieve Subdocument |
| ✓ Submit Document | ✓ Remove a Subdocument |
| ✓ Retrieve Initial Document | ✓ Update a Subdocument |
| ✓ Export Subdocuments | ✓ Retrieve the Version of a Subdocument |
| ✓ Retrieve the Business Signing Certificate | ✓ Retrieve Thumbnail |
| ✓ Update the status of a user message | ✓ Retrieve Document Version |
| ✓ Retrieve a list of subdocuments | ✓ Retrieve Document |
| ✓ Create a Subdocument | |

Create document:

Clerk chooses to add a new document to an existing case instance.

He first manages the participants for this new document: he selects some participants in the list of available participants. The available participants are the initial participants (defined in the CSN) or a subset (of this list). There are three types of documents in this context:

- Setting Participants – when a case is created a document has to be submitted with the list of participants of the case. Here, the set of participants are selected from the list of all participants (defined in the CSN), which have the case type in their scope.
- Documents with all participants from the case – there are document types, which are always sent to all of the participants of the particular case.
- Documents with a subset of the participants of the case – there are document types, where the receiving participants are selected from the list of the participants of the particular case.

For example, a P2000 will present the complete list of CSN participants (first message) that have in scope P_BUC_1 case type, but a P10000 will present the list of the participants involved in the current case.

He is presented with the document pre-filled with the process specific context information (e.g.: demographics) – data transposition.

Clerk manually updates the SED content and saves it. When the SED is created the first version of the document is created.

He can optionally add attachments to the document.

Update document:

Clerk selects the option of updating an existing document.

He is presented with the document filled with the information provided during last create or update.

Clerk manually updates the SED content. Upon this operation, the version of the SED is updated.

Delete document:

Clerk selects the option of deleting an existing document.

The document disappears from the available document list.

NOTE: the operation can be performed only if the document has not been sent yet.

Send document:

This use case contains the sending of the SED and the sending of the attachment, which can be two different processes. See message exchange for details.

This use case implements these requirements. If this SED is sent to one participant one user message is created and exchanged (from the RINA application). If the SED is sent to multiple participants, multiple user messages are created and exchanged for the same conversation (one user message for each participant as part of the exchange of the specific message instance). When for the same SED an update is performed, a new conversation is registered and one or more user messages are created and exchanged based on the list of the recipients.

The send action is triggering BMI Service that will perform the actual sending in an asynchronous way.

4.2.4 Manage document/subdocument attachments

Managing attachments involves attaching/removing/updating/downloading an attachments to/from an existing document/SED or from a subdocument of a document.

The functions related to Case attachments are as follows:

- Submit attachment on a document
- Retrieve attachment of a document
- Delete attachment of a document
- Update the Medical Information flag of a document attachment
- Send an email with attachment
- Submit attachment on a subdocument

- Retrieve attachment of a subdocument
- Delete attachment of a subdocument
- Update the Medical Information flag of a subdocument attachment

4.2.5 Manage document comments

The user must be able to read, create or delete comments related to a document.

Read document comment:

Clerk selects a document.

The populated comments related to that document are read.

Create document comment:

Clerk selects a document.

Clerk adds a comment to the selected document.

The timestamp and the username must be automatically associated to the comment.

Delete document comment:

Clerk selects a document.

Clerk deletes one of the comments related to the selected document.

4.2.6 Case assignment

The user must be able to assign a case to the users or groups part of the local organisation.

User selects a case.

User selects case assignment.

User configures assignment by updating one or more fields like importance, urgency and, for each actor, the users or groups assigned for this case.

User can request to be assigned to a case

User runs the update action.

4.2.7 Manage case comments

The user must be able to read, create or delete comments related to a case.

Read case comment:

Clerk selects a case.

The populated comments related to that case are read.

Create case comment:

Clerk selects a case.

Clerk adds a comment to the selected case.

The timestamp and the username must be automatically associated to the comment.

Delete case comment:

Clerk selects a case.

Clerk deletes one of the comments related to the selected case.

4.2.8 Manage case attachments

Managing attachments involves attaching / detaching file to / from an existing case.

The functions related to Case attachments are as follows:

- Submit attachment on a case
- Retrieve attachment of a case
- Delete attachment of a case
- Update the Medical Information flag of a case attachment
- Send an email with attachment

4.3 Notification Management Function

4.3.1 Notification Centre (Configuration Management)

Only admin's users have access to the Notification Centre that allows the configuration of notifications based on their types. There are two different types of notifications: ones corresponding to business Exception and the rest to other operations. Among the other operations, individual notifications can be generated when messages get either successfully delivered, validated erroneously or expired to each receiving case participant. The events are triggered in such situations when the positive or negative acknowledgments (ebMS receipts or errors) are processed.

The Notification Centre allows admins to set the retention and notification period (in days) for a particular Business Exception's notification. The retention period is the period after which a business exception is triggered based on the pending message. On the other hand, the notification period is the period after which a notification is sent to the admin informing him/her that a business exception would be triggered. In case each SED of a particular conversation (message sent to multiple case participants) has been associated to successful exchanges, it means that the specific conversation has been completed successfully.

The admin can control whether the clerks can or cannot receive a specific type of classical notification and if he/she (the admin) intends to receive a particular type Business Exception's notification or not.

Information about the way the notifications can be configured can be found under the EESSI CPI Reference document (under section 7).

4.3.2 Notification Management

The notification management function covers all aspects of case notification in RINA.

The following figure (Figure 8) shows the use case diagram for this function:

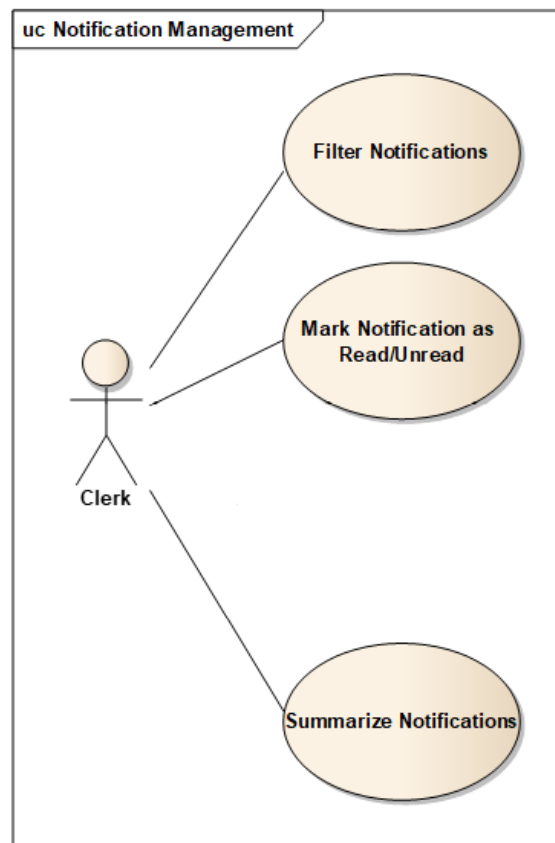


Figure 8. Notification management function use case

4.3.3 Filter notifications

Every time an event condition is fulfilled, RINA will have to notify the assigned clerks (either authorized or non-authorized) and supervisors of the case.

Examples of events are presented below:

- New message arrived;
- New case arrived;
- Message update;
- Message delivered;
- Message delivery error;
- New case assigned;
- New counterparty case was created;
- Etc.

The notifications' events have associated a severity level that could be error, warning or information.

The user should have the possibility to filter the notifications based on notification's type and status and also should be able to navigate into the notifications' timeline.

4.3.4 Mark notification as read/unread

In this use case the user can mark a received notification as read or unread so that the next time the notifications are received the read/unread state is preserved.

4.3.5 Summarize notifications

The user must be able to receive anytime in every module of the application, updates about notifications and he must be able to visualize the number of unread notifications and the number of active errors, warning or informative notifications.

In this use case, the user receives a summary of the notifications in the form:

- The list of error notifications;
- The number of warning notifications;
- The number of information notifications;
- The number of unread notifications.

Also it is possible to receive a consolidated summary that contains only the number of errors, warnings, unread notifications.

4.4 Alarms management function

The Alarms management function provide access to the settings related to operations on Alarms such as:

- Retrieve the alarms for a case
- Set the alarm
- Clear alarm

4.5 Application Profile management function

The application profile management function manages all the settings for the RINA implementation for a respective institution and also manage all the settings related to the communications between various RINA implementations. The settings can be either global and common for the RINA Application, or Tenant specific. Administrator User can:

- Get the Application Profile
- Update the Application Profile
- Retrieving the archiving policies
- Update the archiving policies
- Retrieving the archiving repositories
- Update the archiving repositories
- Retrieving the archiving repositories policies
- Update the archiving repository policies
- Update the Business Exceptions Settings
- Get the Business Exceptions Settings
- Update the IAM Settings
- Get the IAM Settings
- Synchronize LDAP users and groups

- Reassign group affected cases
- Update the message retention policies
- Retrieving the messages retention policies
- Update the Messaging Settings
- Get the Messaging Settings
- Get the NIE subscriptions
- Update the NIE settings
- Delete the NIE subscription
- Update the NIE subscription
- Get Tenant Settings
- Update Tenant Settings

4.6 Application resources management function

The Application resources management function provide access to the settings related to operations on Application resources such as:

- Get the list of application resources

Adds and Institution as a Tenant by the given Institution Id.

The implementation will validate if the Institution can be added.

Institution should exist with the same Id in the IR Repository and should be part of the same Access Point with any other existing Tenant Institution.

4.6.1 Retrieve Tenants' attributes

Retrieves the List of the all the Tenants' attribute.

4.6.2 Retrieve Tenant attributes

Retrieves the Tenant attributes by the given tenant Id.

4.6.3 Delete Tenant

Deletes the Tenant Institution from the application by the given tenant Id.

The Service Implementation will validate if the Tenant can be deleted.

Tenant Institutions with existing cases or users cannot be deleted.

Default Institution cannot be deleted.

4.6.4 Update Tenant

Updates the Tenant with the given tenant Id:

- Enables/disables Tenant or/and
- Make Tenant the default Tenant of the application.

The Service Implementation will only update enabled or/and isDefault field of the Tenant attributes. Other attributes will be ignored.

Service Implementation will also validate if the Tenant is possible to be enabled, disabled or/and becoming the default.

Default Tenants cannot be disabled.

One only Tenant can be default.

4.7 Assignment policies management function

The Assignment policies management function provide access to the settings related to operations on Assignment policies such as:

- Create a case assignment policy
- Get the case assignment policies
- Get the case assignment policy
- Delete a case assignment policy
- Update a case assignment policy
- Add a target to an assignment policy
- Update a target with a list of assignment policies
- Remove a target from an assignment policy

Each Tenant has its own independent Assignment Policies. Therefore, all the above function use cases are performed in relevance with a specific Tenant.

4.8 Audit logs management function

As the audit trail is a fundamental part of EESSI system, particularly useful for assisting with information in case of system failure.

The audit logs management functions ensure the API access to the audit logs by retrieving a list of audit logs details and retrieving the audit logs of specific time slots.

The audit logs can be searched by specifying a type and an id; both of them in the same time.

4.9 Pending Messages and Business Exceptions

The Pending Messages and Business Exceptions management function provide access to the settings related to operations on Pending Messages and Business Exceptions such as:

- Retrieve the business exceptions details
- Submit a business exception for a pending message
- Retrieve the business exceptions time slots
- Retrieve the SED content of the business exception
- Retrieve the rejection SED content of the business exception
- Retrieve the pending messages details
- Retrieve the pending messages time slots
- Retrieve the SED content of the pending message

4.10 Sanity test management function (buckets definitions)

The checkbuckets definition management function is used in order to do sanity tests over local RINA implementation (portal).

4.11 Check Definitions

The Check Definitions management function provide access to the settings related to operations on Check Definitions such as:

- Update a check definition
- Get all check definitions

4.12 Entities management function

The entities management function provide access to details about different entities or RINA implementations such: id, name, country codes, access point details, location, assigned BUCs, etc.

4.13 Files

The Files management function provide access to the settings related to operations on Files such as:

Uploads a file to the server and returns its full path on the server

4.14 Keystores management function

The keystores management function provide the means to manage the private and public certificates (including TLS certificates) for the local RINA implementation.

4.15 Authorisation management function (process definitions assignment)

The authorisation management function allows the assignment policies definition which describes the users / groups and what roles they should have in certain conditions. Also through this functions the association of them with entire definitions and processes groups can be made. This can be done with certain process definitions or punctually with certain process definitions actors.

4.16 Physical artefact management function (resources)

The physical artefact (resources) management function provide manage the settings such as update & retrieve operations related to the local RINA implementation application resources.

4.17 Sectors management function

The sectors management function provide access to the details related to sectors of a local RINA implementation

4.18 Synchronisations management function

The Synchronisations management function provide access to the settings related to operations on ApplicationProfile such as:

- Submit Common Data Model Request Document
- Retrieve Initial Common Data Model Request Document
- Submit Institution Repository Request Document
- Retrieve Initial Institution Repository Request Document

4.19 Technical logs management function

The technical logs management functions provide the access to the technical log operation such as retrieving the technical log details for day and retrieving the technical logs' time slots.

4.20 User Management Function (Groups)

The user (groups) management function provide access to the settings related to operations on user groups such as: retrieving details related to user, group, user or group details, create/ update/ delete user or groups.

4.21 User profile management function

The user profile management function provide access to the settings related to operations on user profile such as retrieving or updating the user profile.

Additionally, through the "field chooser" service contract function this provide access to settings such retrieving or updating the field choosers.

4.22 Vocabularies management function

The vocabularies management function provides the settings to vocabularies operations (i.e. retrieving concepts of a specific vocabulary type).

4.23 Case Management Scheduling/ Planning

The Case Management Scheduling/Planning (Calendar) allows clerks to create/ update and delete activities. Notifications for a particular clerk are also visible in the calendar.

4.24 Configurations

The configurations functionality provides API methods for changing various configurations object (e.g. case counter settings, LDAP synchronization settings, etc.).

We differentiate between two types of configurations (besides the well-known application profile, or messaging settings):

- Application global – this configurations are valid for the whole application, not depending on the tenant institution. I.e. they will be valid and used for every tenant institution.
- Tenant Institution specific – this configurations are specific for every tenant institution. Changing the values for one tenant institution will not influence the settings for another one in any matter. Calls to update/create/delete this kind of configurations usually must contain the tenant institution ID for identification

4.25 Portal content function

Portal content function includes:

- UI translations – provides all the messages displayed by the portal interface for a specific language;
- SED metadata – provides the portal form metadata content for every SED and version.

5 Subscription Service (WebSocket)

As explained earlier considering the complexity of the business use cases involved in EESSI, the list of actions returned in the case meta-data is dependent on the actual case state. This means, that two subsequent calls to get case meta-data can result e.g. in a different set of actions that can be executed.

There are two options how the CPI client should be constructed:

- Periodically pull the case meta-data and update the GUI, or other components that are dependent.
- Listen to the notifications, which the server is sending and handle them.

In this chapter we will show how one can make use of the WebSocket STOMP messages and subscribe to the notifications.

5.1 Sample Scenario

To demonstrate why there is a need to always have the most recent metadata of each case, take a look on the sample scenario shown on the figure below (Figure 9):

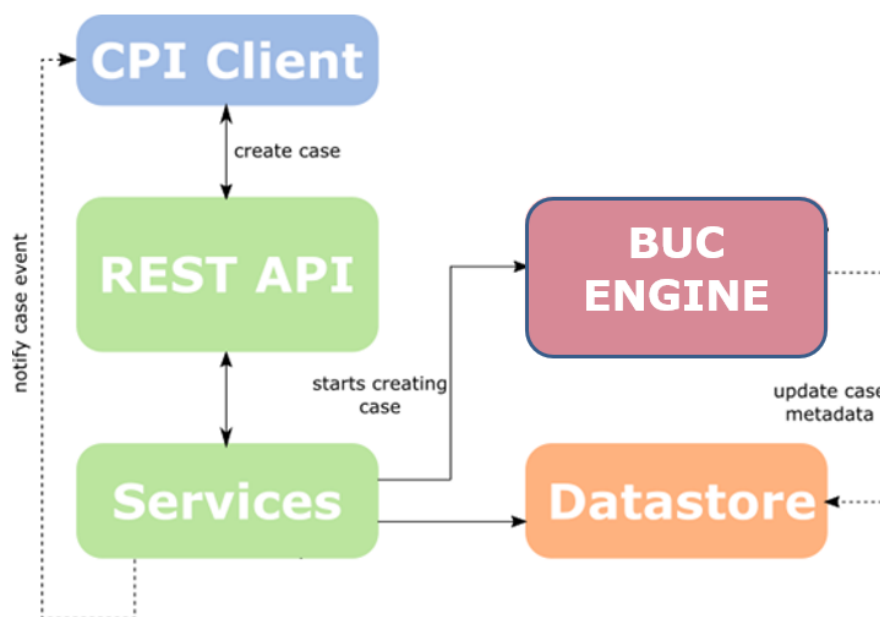


Figure 9. Logical representation of the connection of the processing components

The processing in the BUC engine runs synchronously. A call to the CPI Rest method to create a case returns a caseId after it has triggered the case creation in the BUC engine. An instant call to obtain the case meta-data would probably result in an exception, because the case meta-data would not be found as the processing does not need to be done at that precise moment. The dotted lines represent the synchronous behaviour, whereas the full lines show the process of a single HTTP call.

5.2 Notification format

The payload contained in the notification message is always a JSON formatted data and have to following structure:

```
{
  "description": "description of the notification",
  "progress": {
    "completed": required (true=done),
    "text": optional (the current step name)
    "error": optional (an error message),
    "value": percent, optional (maybe unknown),
    "crtStep": optional (assume 1 if missing),
    "stepCnt": optional (steps count can be unknown),
  }
}
```

Based on the attributes, it is possible to implement a progress bar, show what is actually happening, etc. An important attribute is the “completed” flag. This says, if any action that has been started, has completed or not (E.g. when creating a case, the client application could receive more notification with this flag set to false and then one, which has it set to true).

This format implies two options of how the client application can react:

- React to every notification (recommended) – even if the notification’s flag “completed” is false, the client application would update the case meta-data. In this case, the client application would react faster.
- React to only notifications with the “completed” flag set to true – this results in longer waiting time until the client application updates.

5.3 Subscribing to the case notifications

The server provides a notifications sent through the WebSocket connection to the subscribed clients. Following is the WebSocket Server URI, which should be used when connecting to the server:

```
ws://localhost:8080/eessiRest/events/websocket
```

After the connection has been established, the clients can subscribe to the case notifications. Once we have a valid case ID we can subscribe providing the following Stomp Headers:

```
Id: case-{caseID}
Destination: /topic/cases/{caseId}
```

Now (depending on the WebSocket client library), we can listen to the push notifications, which are sent every time when something changes.

This should usually be the trigger to update the case meta-data and react to the changes. The following sequence diagram (Figure 10) demonstrates the logic:

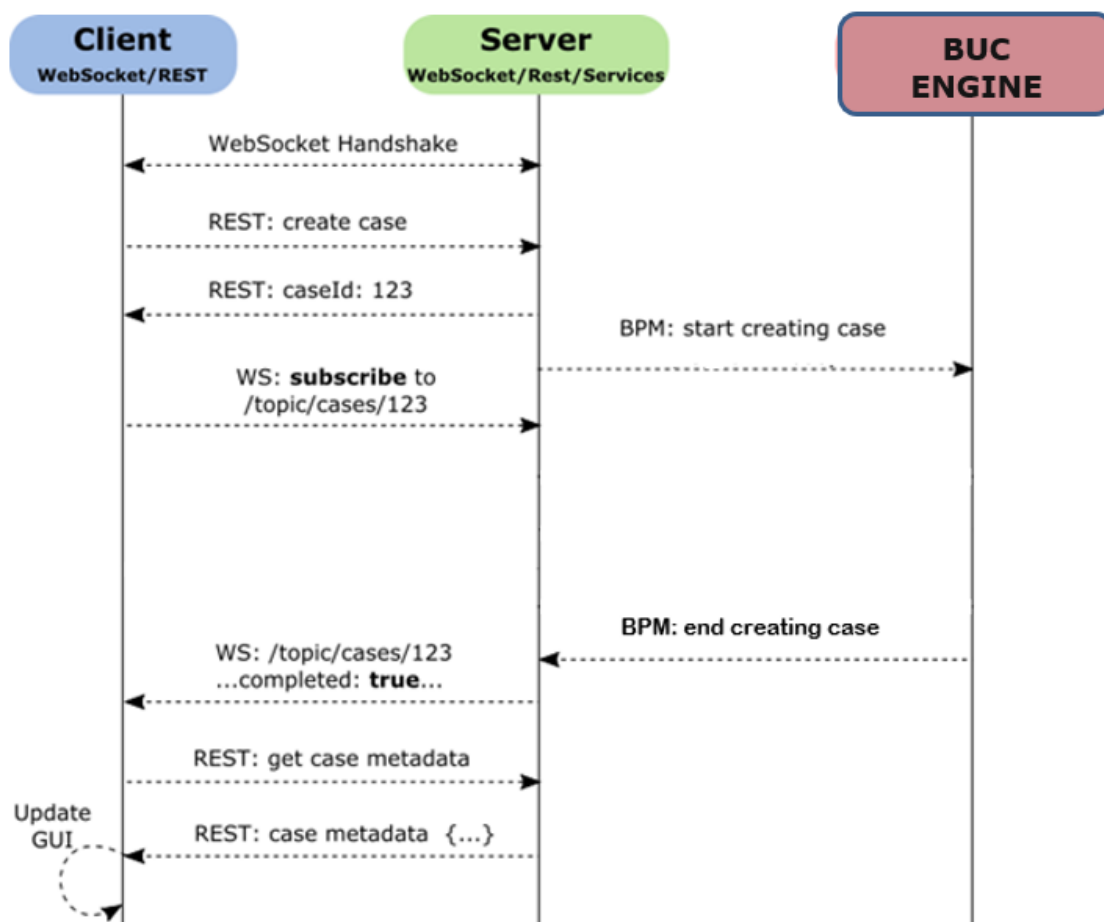


Figure 10. Sequence diagram for receiving notifications through subscriptions

Every subscription is saved on the server with its ID and the timestamp. After successfully processing each event and the corresponding transaction is committed the server will push the notification. Therefore, the client application needs to update the server and send a specified STOMP message to update the subscriptions. This message contains a JSON array of subscription IDs which should be kept alive. Following is an example of such a STOMP message:

```

Destination: /app/subscriptions
Payload: [{"id":"subscriptionID1"}, {"id":"subscriptionID2"}]
    
```

6 Behaviour - state machines

6.1 Case state machine

The case state machine is used to keep track of the case state on either the case owner and the counterparty/ies sides:

- **Open** – In this state the sender party (case owner) creates the case and the receiver party (counterparty) opens the case for the first time. After all the work is done on this case the involved participants may close it. If other participant should be responsible for this case from now on, an existing participant forwards the case. If the case owner is not interested anymore in the case he deletes the case. The case can be deleted if no SEDs have been exchanged; When deleted, a case is removed from the system and it has no specific status. This action cannot be reversed and the Case cannot be undeleted.
- **Closed** - The participant closed the case, from this state the case can be opened again; also un-archiving a case which was closed will bring the case from archived back to the closed status;
- **Removed** - The case has been forwarded to another participant or own participant has been removed and is no longer part of the case, in the latter case the case will be read-only;
- **Archived** – The case is archived either automatically by the system following normal archiving process (following case closing, case forwarding, or case removed participant - according to the defined timers for respective case's BUC Type), or either manually by the user.

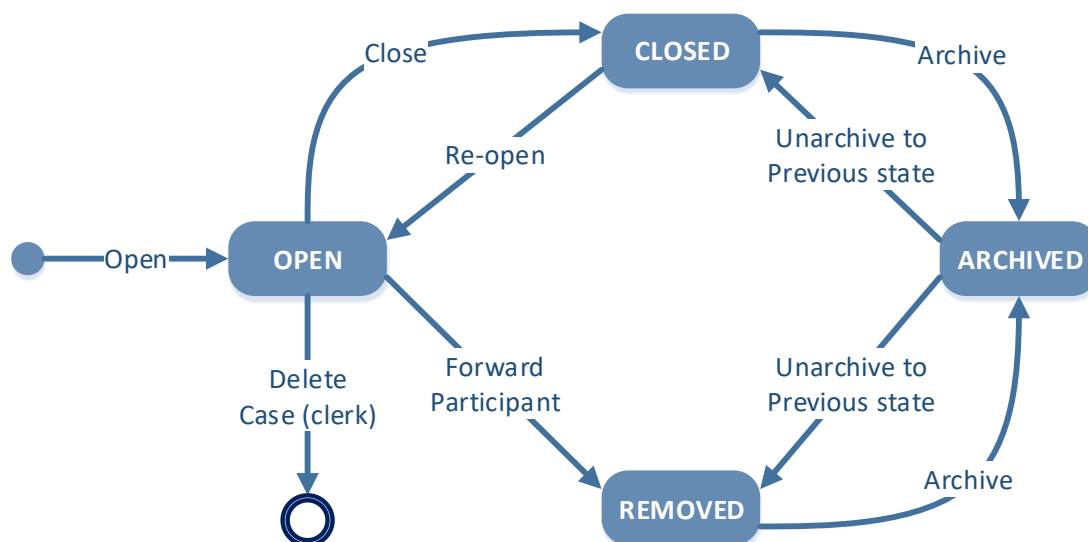


Figure 11: Case state machine

6.2 Document sender state machine

The document state machine is used to keep track of the document state on the sender side:

- **Empty** – This is the Empty state where the document has just been initialized and has no content. Basically there is no SED data, only document metadata is created (e.g. participants, document type, case id, status, version, etc.);
- **New** - In this state the user has a new document (draft) with the relevant SED fields populated (the difference from the previous state is that the document has in addition the SED content); from this state it can be sent or edited again;
- **Sent** - The document is sent, from this state the document can be updated, cancelled or forwarded;
- **Updated** - The document has been updated, from this state the document can be sent back or cancelled;
- **Cancelled** - The document has been cancelled, this is the final state.

NOTE: The document has been invalidated by the sending institution, by using the corresponding document related to the corresponding administrative BUC.

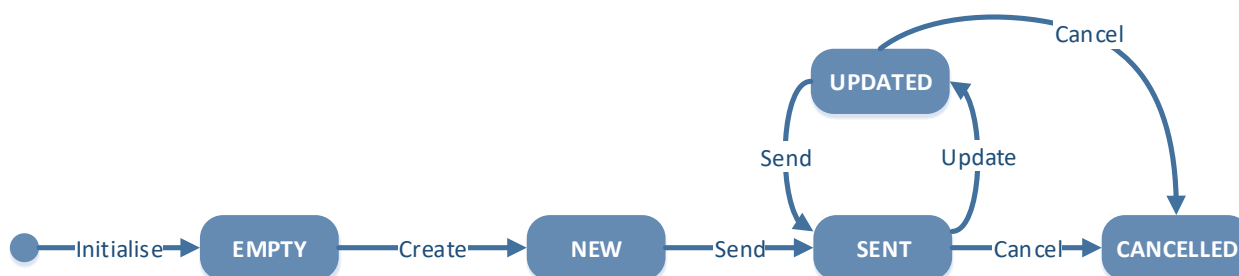


Figure 12. Document sender state machine

6.3 Document receiver state machine

The document state machine is used to keep track of the document state on the receiver side:

- **Received** – The document has been received from the counterparty institution;
- **Cancelled** - The document has been invalidated by the sending institution, by using administrative BUC.



Figure 13. Document receiver state machine

7 Reference Documentation

The reference documentation is located in a separate document ("EESSI - RINA - Case Processing Interface (CPI) - Reference"), which is generated from the code and contains the exact description of the REST calls for each resource. It also contains the description of the model objects, generated from the source codes (service and data contracts).

For testing purposes the application also has a Swagger interface which allows to generate sample values. This interface can be also accessed through a Web Browser under the following (relative) URL: <http://<FQDN>/eessiRest/swagger-ui.html>. In order to get this achieved, the access would have already been granted to the user through the same web browser session (as per section 3.3) and the token has not yet been expired. Last but not least, in order to be able to perform CPI call operations through the Swagger interface via the web browser, it is necessary to provide a valid value of the XSRF token through the relevant section parameter (see Figure 14).

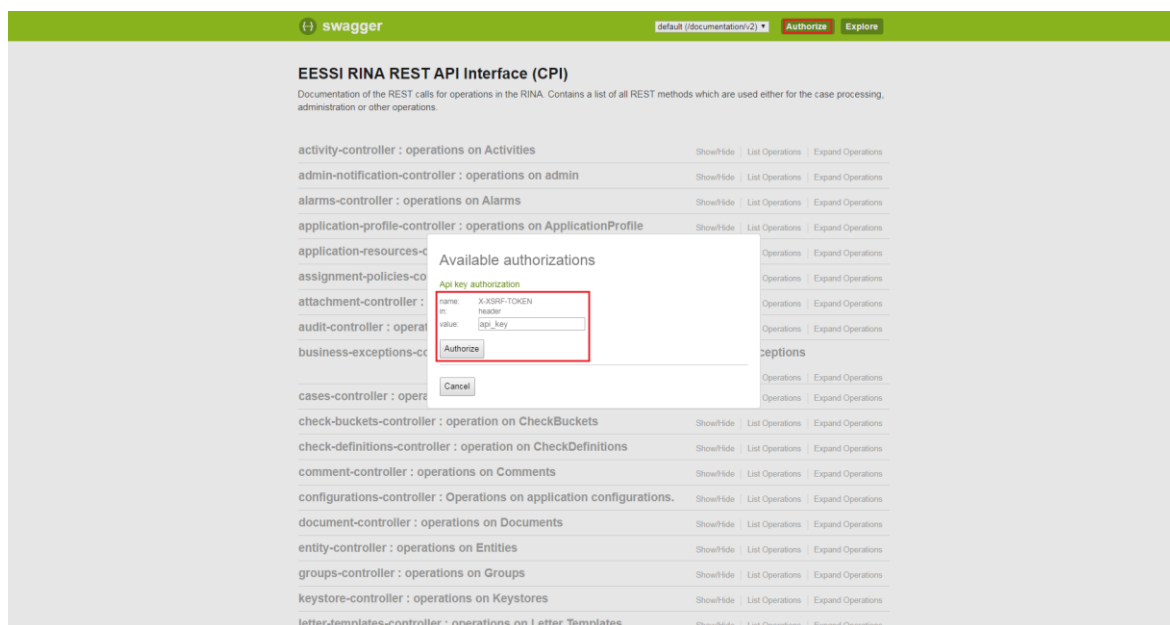


Figure 14. Swagger interface through web browser

CPI Authentication Endpoint

This endpoint is used by CAS in order to authenticate RINA users.

The endpoint uses BASIC authentication and only one set user can make requests on this endpoint. The same username and password are saved on configuration files on both CAS and CPI.

/user-auth

parameter: user credentials

returns: success code if the user credentials are valid or one of the following errors:

Account not found. Not found exception

Account of the user is deleted
Password for the user was incorrect
Account for the user is disabled

Listener for APclient

This endpoint implements the APListener interface and is used by APClient to inform CPI about new message, update message status and update message signature events.

The endpoint uses BASIC authentication and only one set user can make requests on this endpoint. The same username and password are saved on configuration files on both APClient and CPI.

/received

parameter: message json

returns: success code if the message is received and persisted correctly or one of the following errors:

Duplicated pending message for message
Failed to add pending message for message
An error occurred while loading the message content

/signature-update

parameter: message signature json

returns: success code if the signature has been received and persisted with success or an server error code otherwise

If the correspondent message is not found a pending message status entity will be created. Periodically DailyReview process will check again if the message exists and if this is true, it will update the signature of the message

/status-update

parameter: message status json

returns: success code if the status has been received and persisted with success or a server error code otherwise

If the correspondent message is not found a pending message signature entity will be created. Periodically DailyReview process will check again if the message exists and if this is true, it will update the status of the message if no receipt is received from message receiver

ANNEX - Differences of CPI Implementation between RINA 5.x (EESSI 2019) and 6.x (EESSI 2020)

In this ANNEX is provided a detailed description of the differences between the CPI API implementation of RINA 5.x (delivered within EESSI 2019) and RINA 6.x (delivered within EESSI 2021) version.

The differences are given per implemented method and the methods are grouped per module for better navigation (see Table 1 to Table 16).

Document Controller		
Method description	Changes in RINA 6.x	Backwards compatible
Method: retrieveDocumentDetails URL path: /Cases/<caseId>/Documents/<documentId>/Details http request method: GET	RINA 6.x changes the returned type from DocumentDetailsDto to DocumentDetailsRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: submitDocument URL path: /Cases/<caseId>/Actions/<actionId>/Document Http request method: PUT	RINA 6.x changes the returned type from Map<String, Object> to ValidationResultDto, however this is transparent to the user, as this generates the same JSON.	YES
Method: updateSubdocument URL path: /Cases/<caseId>/Documents/<documentId>/Subdocuments/<subdocumentId> http request method: PUT	RINA 6.x changes the returned type from Map<String, Object> to SubdocumentRevisedDto, however this is transparent to the user, as this generates the same JSON.	YES
Method: addSubdocument URL path: /Cases/<caseId>/Documents/<documentId>/Subdocuments/<subdocumentId> http request method: POST	RINA 6.x changes the returned type from Map<String, Object> to SubdocumentRevisedDto, however this is transparent to the user, as this generates the same JSON.	YES
Method: exportSubdocuments URL path: /Cases/<caseId>/Documents/<documentId>/Batch http request method: GET	RINA 6.x changes the URL path to "/Cases/<caseId>/Documents/<documentId>/Batch", From "/Cases/<caseId>/Documents/<documentId>/Batch/<fileType>" in 5.x. File type is redundant, since only the XML file type is supported.	NO
Method: importSubdocuments	RINA 6.x changes the URL path to /Cases/<caseId>/Documents/<documentId>/Batch from /Cases/<caseId>/Documents/<documentId>/Batch/<f	NO

<u>URL path:</u> /Cases/<caseId>/Documents/<documentId>/Batch <u>http request method:</u> POST	ileType> in 5.x. File type is redundant, since the only file type that is supported is XML.	
--	---	--

Table 1: Changes in Document Controller of CPI API

Cases Controller		
Method description	Changes in RINA 6.x	Backwards compatible
Method: restoreCase	removed	NO
Method: getCase URL path: /Cases/<caseId> http request method: GET	RINA 6.x changes the returned type from CaseDto to CaseRevisedDto. CaseRevisedDto has a new field: caseType (string). CaseRevisedDto has a field <i>actions</i>, which is a list of ActionRevisedDto objects. In this object the field <i>operation</i> is an enumeration which had the following values changed: • Import_Subdocument changed to Import_Subdocuments • new value CreateCase • new value Attachment_Removed • new value ForwardCase • new value CreateChild • new value CreateReply • new value Cancel • new value CancelReceive • new value Receive • new value ReceiveUpdate • new value ReceiveReply	NO
Method: getCaseByBusinessId URL path: /Cases/ByBusinessId/<businessId> http request method: GET	RINA 6.x changes the returned type from CaseDto to CaseRevisedDto. CaseRevisedDto has a new field: caseType (string).	YES ³
Method: getCaseByInternationalId URL path: /Cases/ByInternationalId/<internationalId> http request method: GET	RINA 6.x changes the returned type from CaseDto to CaseRevisedDto. CaseRevisedDto has a new field: caseType (string).	YES ³

³ REST API clients must support the extension of input or return structures with new fields, provided the existing fields remain unchanged

Method: getCaseAssignment URL path: /Cases/ByInternationalId/ <internationalId> http request method: GET	RINA 6.x changes the returned type from CaseAssignmentDto to CaseAssignmentRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same	YES
Method: assignCase URL path: /Cases/<caseId>/Assignm ent http request method: PUT	RINA 6.x changes the body type from CaseAssignmentDto to CaseAssignmentRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: executeCaseAssignment Action URL path: /Cases/<caseId>/Assignm ent/Actions http request method: POST	RINA 6.x changes the body type from CaseAssignmentDto to CaseAssignmentRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES

Table 2: Changes in Cases Controller of CPI API

Activity Controller		
Method description	Changes in RINA 6.x	Backw ards compa tible
Method: getActivitiesForUser URL path: /Activities/AllActivities http request method: GET	RINA 6.x changes the returned type from List<ActivityDto> to List<ActivityRevisedDto>. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: createActivity URL path: /Activities/Activity http request method: POST	RINA 6.x changes: - the body type from ActivityDto to ActivityRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same. - the returned type from List<ActivityDto> to List<ActivityRevisedDto>. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: updateActivity URL path: /Activities/Activity/<id> http request method: POST	RINA 6.x changes: the body type from ActivityDto to ActivityRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES

	the returned type from ActivityDto to ActivityRevisedDto. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	
--	--	--

Table 3: Changes in Activity Controller of CPI API

ApplicationProfile Controller		
Method description	Changes in RINA 6.x	Backwards compatible
Method: synchronizeLdap URL path: /ApplicationProfile/IAMSettings/LdapSynchronization/<institutionId> http request method: POST	RINA 6.x changes the return type to void from string.	NO

Table 4: Changes in ApplicationProfile Controller of CPI API

Groups Controller		
Method description	Changes in RINA 6.x	Backwards compatible
Method: deleteGroup URL path: /Identity/Group/<groupId> http request method: DELETE	RINA 6.x changes the return type to string (returns group id) from void.	YES

Table 5: Changes in Groups Controller of CPI API

Keystore Controller		
Method description	Changes in RINA 6.x	Backwards compatible
Method: updateBusinessAliasPassword URL path: /Keystores/UpdateBusinessAliasPassword http request method: POST	RINA 6.x changes the body type from Map<String, Object> to CLIENTBusinessKeyStorePassword, however this is transparent to the user, as this generates the same JSON.	YES

Table 6: Changes in Keystore Controller of CPI API

LetterTemplates Controller

This controller was removed in RINA 6.x as this functionality was not used.

Table 7: Changes in LetterTemplates Controller of CPI API

Notifications Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: retrieveNotificationsDetails URL path: /Notifications http request method: GET	RINA 6.x changes the returned type from List<NotificationDto> to List<NotificationRevisedDto>. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: retrieveNotificationsDetailsForIds	This method was removed from RINA 6.x as it was not used.	NO
Method: updateNotifications	This method was removed from RINA 6.x as it was not used.	NO

Table 8: Changes in Notifications Controller of CPI API

ResourceAdmin Controller

Method description	Changes in RINA 6.x	B/W compatible
Method: getResources URL path: /Resources http request method: GET	RINA 6.x changes the returned type from List<ResourceDto> to List<ResourceRevisedDto>. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES
Method: updateResources URL path: /Resources/<resourceId> http request method: PUT	RINA 6.x changes the body type from List<ResourceDto> to List<ResourceRevisedDto>. Both DTOs have the same fields and although some field types changed, the JSON response is the same.	YES

Table 9: Changes in ResourceAdmin Controller of CPI API

Subscriptions Controller

This controller was removed in RINA 6.x.

Table 10: Changes in Subscriptions Controller of CPI API

Ticket Controller

This controller was removed
in RINA 6.x.

Table 11: Changes in Document Ticket Controller of CPI API

Users Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: updateUser URL path: /Identity/User/<userId> http request method: PUT	RINA 6.x changes the return type from void to UserDto.	NO
Method: registerUser URL path: /Identity/Registration/<userId> http request method: PUT	RINA 6.x URL path: /Identity/Registration http request method: POST RINA 6.x changes the return type from void to string (returns the user id).	NO

Table 12: Changes in Users Controller of CPI API

ApListener Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: onReceivedMessage URL path: /message/received http request method: POST	New endpoint in RINA 6.x	N/A
Method: onMessageSignatureUpdate URL path: /message/signature-update http request method: POST	New endpoint in RINA 6.x	N/A
Method: onMessageStatusUpdate URL path: /message/status-update http request method: POST	New endpoint in RINA 6.x	N/A

Table 13: Changes in ApListener Controller of CPI API

Authentication Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: authenticateUser URL path: /user-auth http request method: POST	New endpoint in RINA 6.x	N/A

Table 14: Changes in authentication Controller of CPI API

Organisation Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: searchOrganisations URL path: /organisations http request method: GET	New endpoint in RINA 6.x	N/A
Method: searchOrganisationsByParam URL path: /organisations/searchByParams http request method: GET	New endpoint in RINA 6.x	N/A
Method: getOrganisationByIdAndCountry URL path: /organisations/<id> http request method: GET	New endpoint in RINA 6.x	N/A

Table 15: Changes in Organisation Controller of CPI API

PortalForms Controller

Method description	Changes in RINA 6.x	Backwards compatible
Method: getTranslations URL path: /forms/translations http request method: GET	New endpoint in RINA 6.x	N/A
Method: getSedMetadata URL path: /forms/metadata http request method: GET	New endpoint in RINA 6.x	N/A

Table 16: Changes in PortalForms Controller of CPI API