



HEALTHeID

eIDAS – OpenNCP
Connector for eHealth

Integration Guide

Document Information:

Document status:	Draft
Approved by HEALTHeID Consortium	
Document Version:	0.10
Author(s):	Klay Rocha, Maximiano Pereira, João Cunha (SPMS), Alberto Zanini (ARIA – former LISPA), Abel Tenera, Carlos Coutinho (Caixa Magica), Andrew Short (AUTH), Giuseppe Burrari (POLITO), Cesare Cameroni (POLITO)
Member State Contributor(s):	SPMS (Portugal), AUTH (Greece), ARIA – former LISPA (Italy), POLITO (Italy)
Stakeholder Contributor(s):	

Summary

1.1 About this document.....	4
1.2 HEALTHeID architectural diagram	4
1.3 Artefacts	4
1.4 Security considerations	6
1.5 HP I/O.....	9
1.6 HeID Client.....	10
1.7 HeID Connector.....	13
1.7.1 Workflow Manager.....	14
1.7.2 eIDAS-HPProxy	26
- Prerequisites.....	27
- Installation.....	27
- Configuration	27
- eIDAS Specific Configuration:	31
New Keystore Setup	32
- Module configuration	34
1.7.3 NCP National Adapter	39
1.7.4 NCP HPProxy	41
1.7.5 Patient I/O.....	46
1.7.6 HeID Connector – Flow Example	52
1.8 NCPeH-A.....	54
1.9 NCPeH-B.....	56
1.10 eIDAS National Adapter	56
1.11 PatientID Resolver	59

Table of revisions		
Date	Comments	Authors
08/08/2019	First draft	Klay Rocha, Maximiano Pereira, João Cunha (SPMS), Alberto Zanini (ARIA – former LISPA), Abel Tenera, Carlos Coutinho (Caixa Magica), Andrew Short (AUTH), Giuseppe Burrai (POLITO)

02/09/2019	Version 0.2 – Revision, comments and new sections 1.3 and 1.4	João Cunha (SPMS)
10/09/2019	Version 0.3 – section 1.10 revised	Alberto Zanini (ARIA – former LISPA)
10/09/2019	Version 0.4 – revision of 1.7.1 and 1.7.2; added Notification Adapter in 1.7.5	Giuseppe Burrari (POLITO)
13/09/2019	Version 0.5 – editorial revision; properties configurations in section 1.7.1, 1.7.4 and 1.7.5	João Cunha (SPMS)
16/09/2019	Version 0.6 – minor revision of section 1.7.2; properties configuration in section 1.7.5	João Cunha (SPMS)
24/10/2019	Version 0.7 – Revision to section 1.7.2; changes to property structure	Andrew Short (AUTH), Cesare Cameroni (POLITO), João Cunha (SPMS)
27/10/2019	Version 0.8 – Transferathon version	João Cunha (SPMS)
13/11/2019	Version 0.9 – Feedback from Transferathon	João Cunha (SPMS)
18/11/2019	Version 0.10 – Feedback from HeID Consortium; Patient I/O improvements	João Cunha (SPMS)

Bibliography		
Id	Document title	Authors

1.1 About this document

This “Integration Guide” has been created to describe the several components developed in the HEALTHeID and give a reference manual to the Member States willing to integrate such components. After a quick view of the architecture implemented in the project, every component is described and details concerning their installation and configuration are provided.

1.2 HEALTHeID architectural diagram

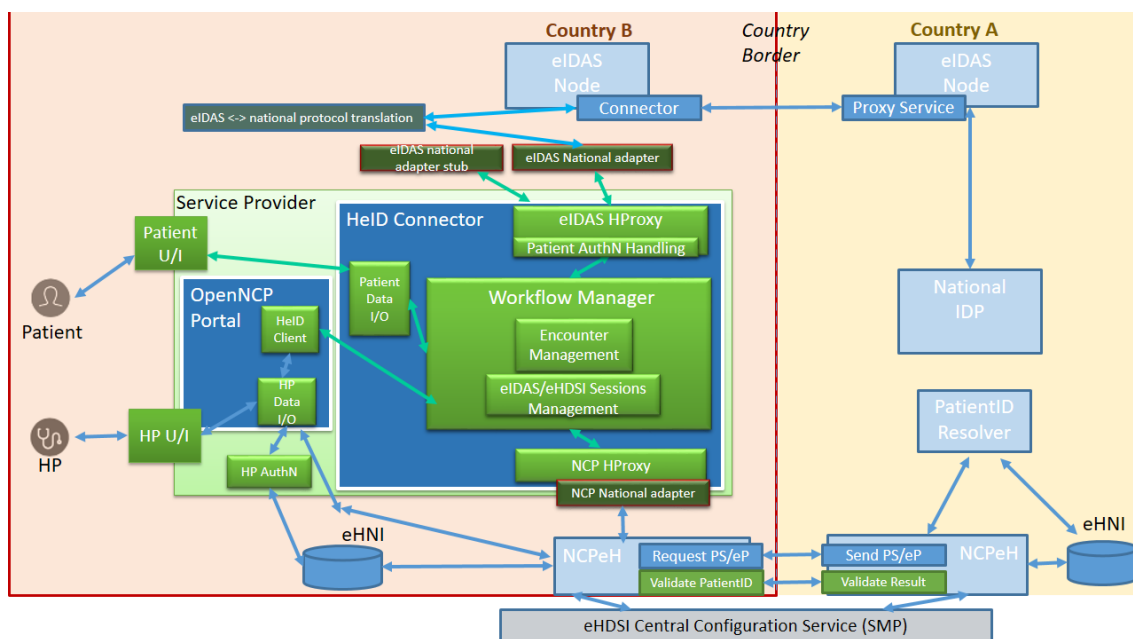


Figure 1 - HEALTHeID Connector architectural diagram

Figure 1 describes both the internal architecture of the HEALTHeID Connector as well as its integration in the overall architecture scenario comprising the NCP and eIDAS worlds, as well as the Service Provider and any potential portal contained within the latter. This architecture and the components identified emerge from the functional requirements identified in deliverable D2.1 HEALTHeID Functional Specification, which we recommend to read in advance.

1.3 Artefacts

HEALTHeID components’ source code is available at CEF Digital OpenNCP Bitbucket in: <https://ec.europa.eu/cefdigital/code/projects/EHNCP/repos/health-eid>

The specific OpenNCP components used in this project are based on version 3.0.0.RC3, so the environment must have this version deployed. For the purposes of

installing the HeID Connector components (see architecture diagram), it's recommended to use a separate Tomcat, preferably version 8.

The following table shows how the different Maven projects materialize into deployable artefacts:

Component	Maven project	Packaging	Deployable (Y/N)	Mandatory (Y/N)
HeID Client	healtheid-client	JAR	Yes, within a portal.	No. Each MS can create its own client. This one is a reference implementation provided for demonstration purposes.
OpenNCP Portal (HP Data I/O)	openncp-portal	WAR	Yes.	No. Each MS can create its own portal. This one is a reference implementation provided for demonstration purposes.
HEALTHeID Connector	healtheid-connector	WAR	Yes	Yes
Patient Data I/O	healtheid-connector	-	No, not as a single component, but contained in the bigger HEALTHeID Connector WAR file.	Yes
Workflow Manager	healtheid-connector	-	No, not as a single component, but contained in the bigger HEALTHeID Connector WAR file.	Yes

	Notification Adapter (not included in the architecture diagram)	healtheid-notification-adapter healtheid-notification-adapter-interface	JAR	Yes, within HEALTHeID Connector.	No, can be replaced by a custom one. This default implementation provides email features.
	eIDAS HProxy	eidas-hproxy	WAR	Yes	Yes
	eIDAS National Adapter	eIDAS National Adapter	WAR	Yes	No
	eIDAS National Adapter Stub	eIDAS National Adapter	WAR	Yes	No
	NCP HProxy	healtheid-ncp-hproxy	WAR	Yes	Yes
	NCP National Adapter	openncp-national-adapter openncp-national-adapter-interface	JAR	Yes, within NCP HProxy.	Yes, either this default implementation or a nationally-developed one.
	OpenNCP Client Connector	openncp-client-connector	WAR	Yes	Yes
	OpenNCP Gateway	openncp-gateway	WAR	Yes	Yes

1.4 Security considerations

The HEALTHeID Connector exposes some internal (between components) and external (to the end-user) endpoints under HTTPS. In order to identify the needs for specific certificates protecting such endpoints, a summary of the components requirements in terms of endpoints and their certificates is presented in the following table.

Note: certificates for OpenNCP components and eIDAS national components are considered out of scope.

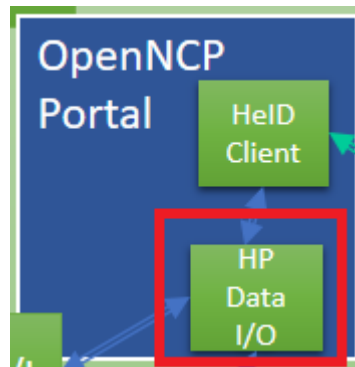
Component	Certificate (Y/N)	Endpoint	Client	HTTPS (1-/2-way SSL/TLS)
HeID Client	No	-	-	-
OpenNCP Portal (HP Data I/O)	No	-	-	-
HEALTHeID Connector	Yes	(see below)	(see below)	(see below)
Patient Data I/O	Yes	Base URL: https://<hostname/ip>:<port> <ul style="list-style-type: none"> • /patientEncounter/patientAcknowledge • /patientEncounter/additionalPatientData 	Patient (end-user)	1-way SSL/TLS
Workflow Manager	Yes	Base URL: https://<hostname/ip>:<port>/healtheid-connector	-	1-way SSL/TLS
		<ul style="list-style-type: none"> • /encounter/createEncounter • /encounter/requestPatientData • /encounter/receiveNotice 	HeID Client	
		<ul style="list-style-type: none"> • /patientEncounter/acceptEncounter/{token} • /patientEncounter/additionalPatientData • /patientEncounter/patientAcknowledge • /patientEncounter/acknowledge 	Patient I/O	
		<ul style="list-style-type: none"> • /heidconnector/acceptPatientAuthN 	eIDAS HProxy	
Notification Adapter	No	-	-	-

eIDAS HProxy	Yes	https://<hostname/ip>:<port>/eidas-hproxy/eidas-hproxy/metadata	eIDAS Node or National Adapter	1-way SSL/TLS
		https://<hostname/ip>:<port>/eidas-hproxy/eidas-hproxy/AuthResponse	eIDAS Node or National Adapter	
		https://<hostname/ip>:<port>/eidas-hproxy/eidas-hproxy/authentication	Workflow Manager	
eIDAS National Adapter	-	-	-	-
eIDAS National Adapter Stub	-	-	-	-
NCP HProxy	Yes	https://<hostname/ip>:<port>/healtheid-ncp-hproxy/ncphproxy/{country-code}	Workflow Manager	1-way SSL/TLS
NCP National Adapter	No	-	-	-
OpenNCP Client Connector	-	-	-	-
OpenNCP Gateway	-	-	-	-

As for the total number of certificates needed, it depends on the deployment choice of the MS. Since the components are loosely coupled, they can be deployed in separate infrastructures (e.g., belonging to different organizations) or all in the same. It is highly recommended that the certificates contain a Subject Alternative Name attribute: although the source code doesn't explicitly perform such check, the used libraries demand the existence of such attribute.

At application level, the HEALTHeID Connector makes use of a signed Json Web Token (JWT) for authorization purposes. More details on its usage and configuration of its symmetric cryptography parameters can be found on the section HeID Connector.

1.5 HP I/O



To take advantage of HEALTHeID-enhanced OpenNCP Portal, the version provided by the HEALTHeID project must be deployed by the country (openncp-portal WAR file).

In order to enable the HEALTHeID features in the OpenNCP Portal, the following properties must be set in the properties database schema (ehealth_properties) of the OpenNCP, in the table EHNCP_PROPERTY:

NAME	IS_SMP	VALUE
HEALTHeID_ENABLED	b'0'	true
HEALTHeID_URL	b'0'	<a href="https://<hostname/ip>:<port>/healtheid-connector">https://<hostname/ip>:<port>/healtheid-connector

- HEALTHeID_ENABLED: enables/disables the HEALTHeID features (true/false)
- HEALTHeID_URL: HEALTHeID Connector services endpoint (HTTPS). The hostname/IP and port are those of the environment where the HEALTHeID Connector runs. The value of this property is communicated to the HeID Client component.

Note: these properties were stored in the ehealth_properties schema of the NCP for ease of use and due to the tight coupling between the OpenNCP Portal and the NCP itself. Other portals are free to store these properties using other mechanisms (e.g., a dedicated schema, a properties file, etc), given that such portals can access them.

This version of the OpenNCP Portal can still be deployed and used in the current eHDSI scenario without adding these 2 properties to the schema: the end-user won't notice any difference.

This version of the OpenNCP Portal expects the certificate chain protecting the HEALTHeID_URL endpoint to be included in the truststore configured in TRUSTSTORE_PATH and TRUSTSTORE_PASSWORD properties of the ehealth_properties schema.

Once the features are enabled, the country page in the OpenNCP Portal must display a radio-button to switch between the current eHDSI scenario (“Search Patient”) or the HEALTHeID one (“Create Encounter”), as depicted in Figure 2. Once in the “Create Encounter” screen, the Portal keeps polling the HEALTHeID Connector every 15s, waiting for new information to be displayed to the HP.

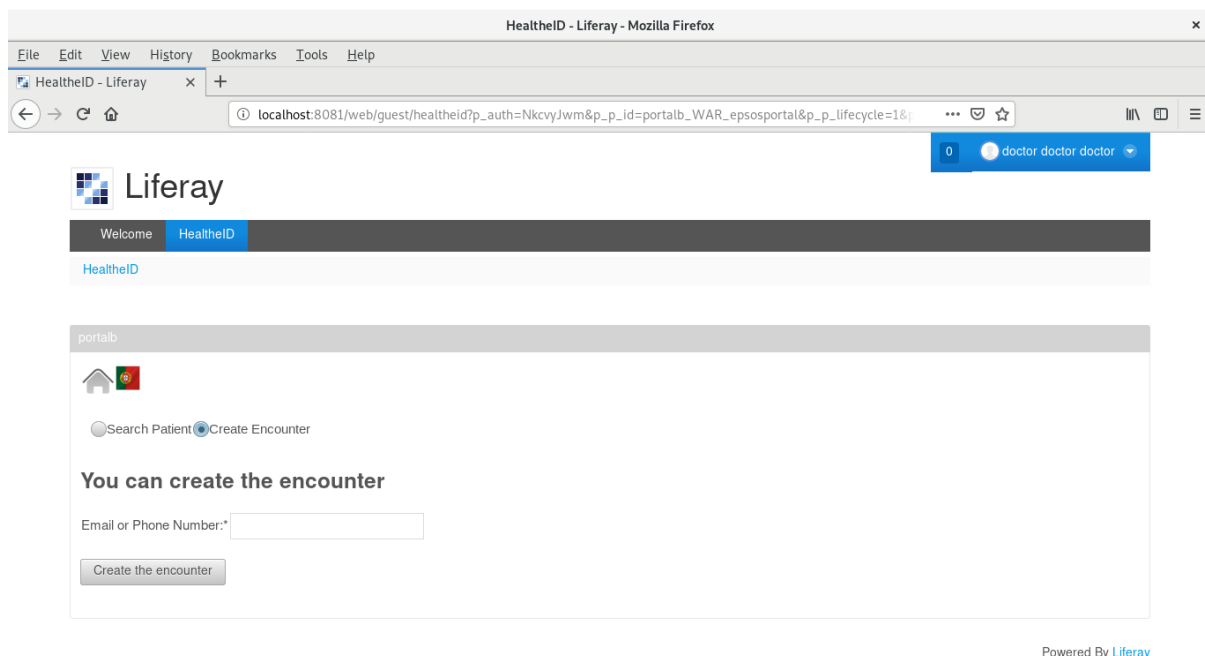
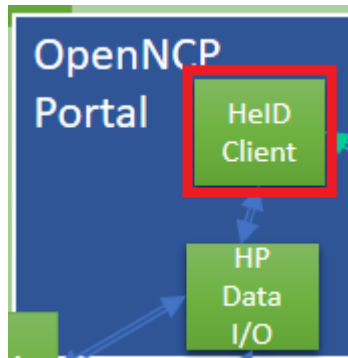


Figure 2 – HEALTHeID-enhanced OpenNCP Portal

1.6 HeID Client



HEALTHeID Connector Client is a component that has the objective to ease the communication with the HEALTHeID Connector services, and it is intended to work not only in an OpenNCP Portal instance, but in another Java environments. Keep in mind the further documentation to avoid compatibility issues. This component is packaged as a JAR file (healtheid-client).

These instructions will present a guide on how to integrate HEALTHeID Client.

Prerequisites: Java 1.8 and Maven.

Installing: pull the project to your working directory, and add the following dependency to POM.

```
<dependency>  
  <groupId>eu.europa.ec.healtheid</groupId>  
  <artifactId>healtheid-client</artifactId>  
  <version>1.0.0-SNAPSHOT</version>  
</dependency>
```

Dependencies

Currently are in use the following dependencies.

Group ID	Artifact ID	Version	Scope
org.springframework.boot	spring-boot-starter	2.1.4.RELEASE	compile
org.springframework	spring-web	5.1.6.RELEASE	compile
org.springframework.boot	spring-boot-starter-test	2.1.4.RELEASE	test
com.fasterxml.jackson.core	jackson-databind	2.9.8	compile
log4j	log4j	1.2.17	compile
com.google.code.gson	gson	2.8.5	compile
org.mock-server	mockserver-netty	3.10.18	test
org.mock-server	mockserver-client-java	3.10.18	test

Keep in mind that incompatibilities may occur with existing dependencies. It is advised to execute "mvn dependency:tree" when integrating HEALTHeID Client when a conflict occurs to check for incompatibilities, and to visit Maven Central to check this dependencies requirements.

Example case of incompatibility: The following error occurred when instantiating a RestTemplate object from spring boot class, after the first integration with OpenNCP Portal, indicating an incompatibility with an existing Jackson Core dependency:

```
| ERROR com.liferay.faces.bridge.context.ExceptionHandlerAjaxImpl -  
java.lang.NoSuchMethodError:  
com.fasterxml.jackson.core.JsonFactory.requiresPropertyOrdering()Z | |:--- |
```

After getting the dependency tree of the project, it was found that the existing jasperreports dependency also imported an outdated version of Jackson Core, so an exclusion was applied.

How to deploy in OpenNCP Portal: to deploy in OpenNCP Portal, follow the instructions in the "Installing" section and run "mvn dependency:tree" in the root of the project. Check for all the dependencies that import Jackson Core, and add the following exclusions to each one:

```
<exclusion>  
  <artifactId>jackson-databind</artifactId>  
  <groupId>com.fasterxml.jackson.core</groupId>  
</exclusion>
```

```
<exclusion>  
  <artifactId>jackson-annotations</artifactId>  
  <groupId>com.fasterxml.jackson.core</groupId>  
</exclusion>
```

```
<exclusion>  
  <artifactId>jackson-core</artifactId>  
  <groupId>com.fasterxml.jackson.core</groupId>  
</exclusion>
```

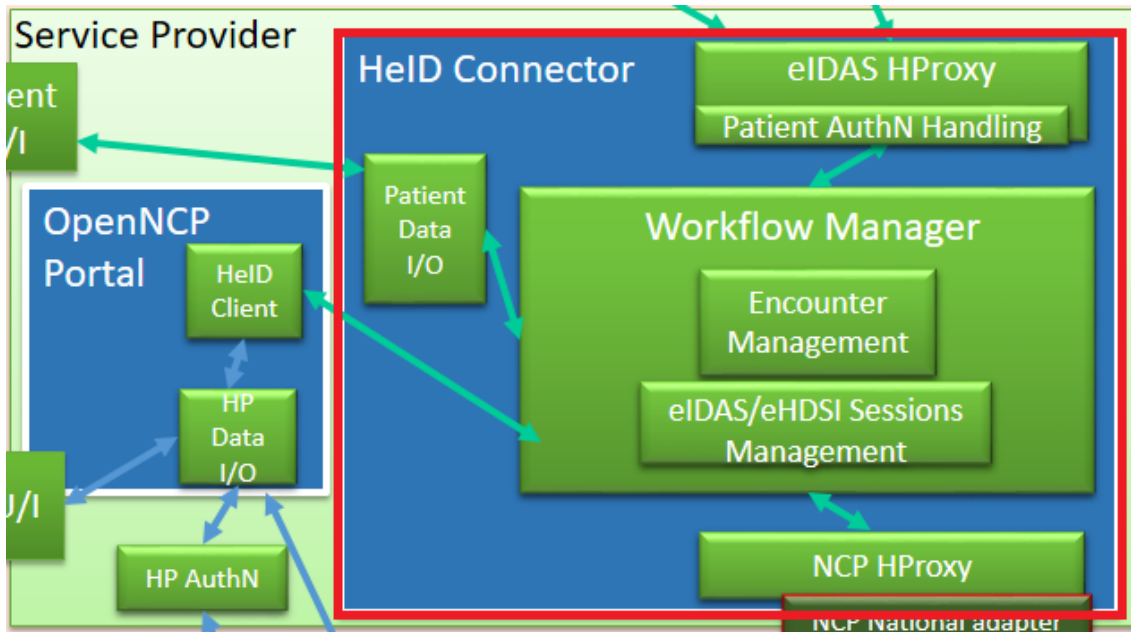
Running the tests: to run the tests, execute "mvn test" on project main directory.

The component is built with:

SpringBoot - Framework used to send messages to HEALTHeID Connector

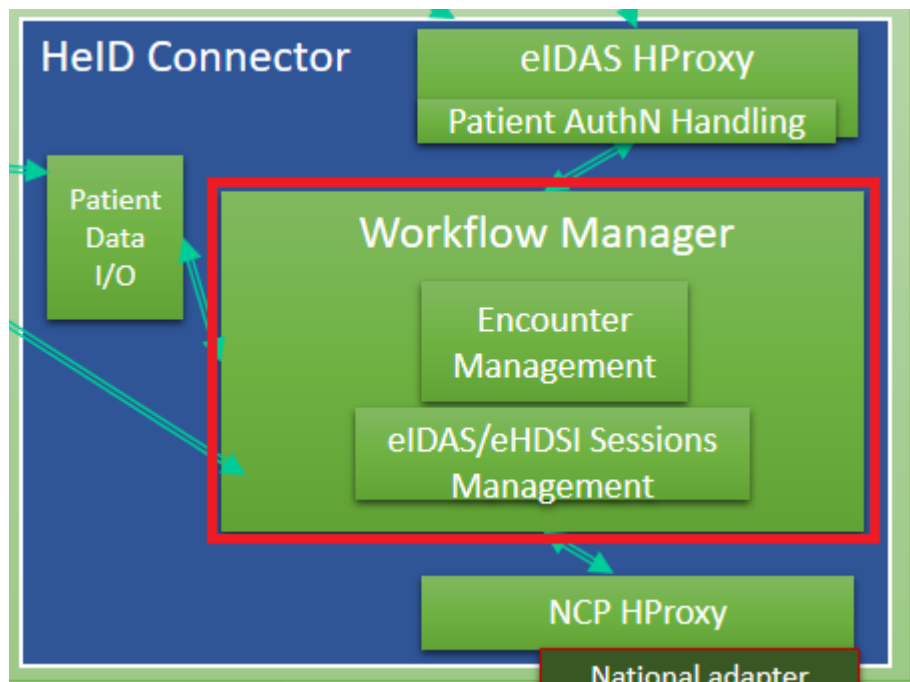
Maven - Dependency Management

1.7 HeID Connector



This section goes into the different sub-components of the HeID Connector.

1.7.1 Workflow Manager

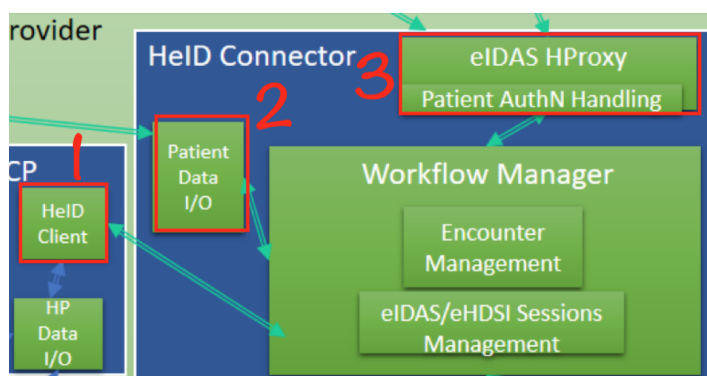


The **Workflow Manager** (WM) handles and orchestrates all the Patient and Healthcare Professional (HP) interactions during the Patient Authentication process, in order to provide a strong identification ensured by the eIDAS

infrastructure. To perform a successful Patient authentication - we assume that Healthcare Professional is already authenticated -, the WM has to interact with three components:

- 1) **HeID Client**, which submits the HP requests towards WM.
- 2) **Patient Data I/O**, that reports Patient decisions and interactions.
- 3) **eIDAS-HProxy**, that generates the eIDAS authN request towards the eIDAS world and then elaborate the eIDAS assertion.

For this reason, the structure of the HeID Connector includes a Spring Controller for each component, respectively the Heid-Client Controller, Patient Controller and eIDAS-HProxy Controller.



HeidClient-Controller handles all the requests coming from the HeidClient component (driven by HP actions). This controller provides three endpoints to satisfy HP requests:

- `encounter/createEncounter`: to create an Encounter between HP and Patient. The encounter is given by a Json Web Token that will be send to both, as HTTP response to the former and via email/sms to the latter (inside the “acceptEncounter link”). Note that the JWT token, also called EncounterID, will be used in the HTTP Authorization Header as Authorization token.
- `encounter/requestPatientData`: used by HeId-Client to check if eIDAS patient data are available. A 204 NO CONTENT will be send as response if data are not yet ready, so the Heid-Client will have to try again later.

- encounter/receiveNotice: through this endpoint the HeID-Client sends notifications about what is happening in the OpenNCP world with the outcome of each process (XCPD, eP-PS, eD). The Workflow Manager is responsible to forward the notice to the Patient.

Some details about the Patient Controller:

- /patientEncounter/acceptEncounter/{token} : called by the Patient when he clicks on the link. The token is the JWT token (EncounterID).
- /patientEncounter/acceptEncounter: redirects the patient to the eIDAS authentication.
- /patientEncounter/patientAcknowledge: called to show the Patient Information Notice (PIN) to Patient.
- /patientEncounter/acknowledge: it accepts the patient acknowledge.
- /patientEncounter/patientConsent: collection of patient consent. (*implemented but not used*)
- /patientEncounter/additionalPatientData: if the eIDAS authentication does not provide all the necessary data, the patient can add them by hand.

The eIDAS-Proxy Controller receives the Patient eIDAS attributes from the eidas-HProxy, after a successful authentication by patient on eIDAS.

- /heidconnector/acceptPatientAuthN : used by eidas-HProxy to communicate the eIDAS map attributes to the workflow manager.

Workflow Manager:

- Creates the encounter (generates the EncounterID between Patient and HP) (EncounterID = JWT Token)
- Validates the JWT token
- Redirects the patient towards eIDAS-HProxy
- Stores the eIDAS attributes for a limited period of time (waiting the HeID-Client data request)
- Contacts the NCP HProxy at the endpoint hostname-hproxy/ncphproxy/{country} to obtain the health-eid country configuration and

maps the eidas attributes with OpenNCP attributes. The {country} parameter is the country value for which the configuration is needed (patient's country)

- Asks and stores the acknowledgement in DB
- Sends notification to patient about what is happening in OpenNCP (XCPD, eP/PS, eD)

The Json Web Token is a secure token used to represent and uniquely identify the encounter between Patient and HP. Note that this token correspond to a string in the following format **aaa.bbb.ccc**, where:

- aaa = JWT header
- bbb = JWT payload
- ccc = Workflow Manager signature = HMAC-SHA256(
 - base64UrlEncode(header) + . +
 - base64UrlEncode(payload),
 - MY_SECRET_KEY)
 = HMAC-SHA256(aaa.bbb, MY_SECRET_KEY)

JWT header, JWT payload and WM signature are base64 encoded.

Example of JWT token (base64 encoded):

```
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiIxYzE0YzE5MC0wZmFjLTRlMTktYjk0Yi0yZGVhbnZg3ODYwZjkiLCJleHAiOiE1NjMyMzYzMjEsInN1YiI6IklUIn0.folhtBB2qH-vyKGOFcivYc54VeAVhg6GtPgaoEi68IA
```

The same token, but decoded:

```
{"alg":"HS256"} {"jti":"1c14a190-0fac-4e19-b94b-2deb787860f9","exp":1563236321,"sub":"IT"} m\^X`k0??
```

The JWT header contains the algorithm used to sign the token (in this case HMAC with SHA-256).

The JWT payload contains a JwtTokenId that is a unique id, an expiration date and a subject attribute that correspond to the Patient Country.

The Workflow Manager can verify the integrity of the token checking the

signature; in practice, it compute the HMAC-SHA256(aaa.bbb, MY_SECRET_KEY) of the token just received, and then checks if this last HMAC is equal to the “ccc” token value. If the two values are equals, the WM knows that this token was generated by itself and then the token is accepted (if not expired); otherwise the token is rejected.

Security Considerations

Please note that the JWT token, used for creating the encounter, is also used as an authentication token by the HeID Client. Therefore the Workflow Manager validates the HeID client requests checking this token. But a problem occurs in the use of this token as authentication header: it is the same for each request performed by the HeID, and it is never changed. This fact makes the connector vulnerable to replay attacks. To avoid this problem, it is advisable to use a NONCE (a number, generally random or pseudo-random, **that has a unique use**) in the authentication header instead of the jwt token. Note also that in our development, we exploit the use of authentication header to validate the request and also to identify the encounter (because the token is the same and corresponds to the encounterID).

Therefore, to improve the current solution, this is what should happen:

1. the workflow manager creates the encounter, using a jwt token as encounterID (not changed)
2. The workflow manager sends the encounterID to HeID client with a **nonce**
3. At the next request, the HeID client uses the **nonce** as authentication token inside the HTTP authentication header, and inserts the encounterID in the HTTP body
4. The workflow manager (WM) receives the message and checks the **nonce**. If the nonce is valid, the WM reads the encounterID from the HTTP body and processes the request. After that, a new **nonce** is sent to the HeID client inside the response; in this way it can use the new nonce in the next request.

This approach should be used also in the interactions with the patient.

Configurations

Some details about the several configurations needed, starting with the HeID Connector:

Prerequisites:

- MySQL 5.x, with a schema named “ehealth_healtheid”
- Apache Tomcat 8.5.x

MySQL database is not mandatory and the user can configure this aspect according to his preferences. It is also possible to use multiple and separate databases according to the Spring JPA configuration. The ehealth_healtheid schema tables will be created upon deployment of this component on Tomcat. Following are the tables that are part of this schema:

- eidas_attribute: stores information of the eIDAS assertion of a specific encounter;
- eidas_ehealth_configuration: stores information about the eIDAS eHealth Configuration needed in a specific encounter;
- encounter: stores encounter-specific information;
- search_maskText_field: stores information about a text field declared in the search mask of a specific encounter;
- search_mask_birth_date: stores information about a birth date field declared in the search mask of a specific encounter;
- search_mask_id: stores information about an identifier field declared in the search mask of a specific encounter;
- search_mask_sex: stores information about a sex field declared in the search mask of a specific encounter;

Go to tomcat_folder/conf/server.xml and add the Global JNDI Resource inside the <GlobalNamingResources> element for the MySQL configuration:

```
<Resource name="jdbc/ConfHeiD" auth="Container" type="javax.sql.DataSource"
    factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
    maxTotal="20" maxIdle="10"
    driverClassName="com.mysql.cj.jdbc.Driver"
```

```

url="jdbc:mysql://localhost:3306/ehealth_healtheid?useTimezone=true"
username="username"
password="password"
/>

```

Then go to `tomcat_folder/conf/context.xml` and define the resource link inside the Context element:

```

<ResourceLink global="jdbc/ConfHeiD" name="jdbc/ConfHeiD"
type="javax.sql.DataSource"/>

```

The following dependency was added for including the MySQL Connector/J driver, in order to obtain the JDBC APIs for communicating with the relational database.

```

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>

```

This component is packaged as a deployable WAR file (`healtheid-connector`). For testing purposes, it can be launched as a standalone Spring Boot JAR file containing an embedded Tomcat.

Its behavior is managed by a set of properties within its self-contained `default.properties` file.

To test the connector in `localhost:8080`, set in the `default.properties` file the “`server.url=localhost`” and “`server.port=8080`”.

To configure JWT Token, specify the “`jwt.secret`” value and the “`jwt.expiration`” value in order to use your custom private secret and to set the expiration time of the token (in seconds).

If you want test the connector without SSL, you have to set `server.ssl.enabled=false` in the `default.properties` file; otherwise you set the property to true and configure the following ssl properties.

Change the component base URL in `default.properties` according with your deployment.

The `default.properties` file contains the following objects:

Property	Value	Description
----------	-------	-------------

server.port	443	Listening Port. This is the port contained in the URL sent to the patient. It's also the port used when the component is run as a standalone JAR file.
server.url	localhost	Url basename. This is the hostname contained in the URL sent to the patient. It's also the hostname used when the component is run as a standalone JAR file.
server.protocol	https	Defines as HTTPS both the URL received by the patient as well as the endpoints exposed when the component is run as a standalone JAR file. It must be "https".
server.ssl.enabled	true	To enable/disable SSL connections. If used, it should be "true". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
security.require-ssl	true	Requires SSL. If used, it should be "true". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store-type	JKS	The format used for the keystore. It could be set to JKS in case it is a JKS file. Only used in the case the

		component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store	/home/user/health-eid/healtheid-connector/keystore/keystore.jks	The path to keystone containing the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store-password	password	Password used to generate the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-alias	healtheid-connector	The alias mapped to the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.tomcat.remote-ip-header	x-forwarded-for	Name of HTTP header from which the remote IP is extracted. If used, it must be "x-forwarded-for". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration

		is provided by the application server where it's deployed.
server.tomcat.protocol-header	x-forwarded-proto	Enable setting the header of incoming protocol. If used, it must be "x-forwarded-proto". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.tomcat.redirect-context-root	true	Whether requests to the context root should be redirected by appending a / to the path. If used, it must be "true". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
jwt.secret	shared-secret-to-sign-and-verify-JWT-token-change-it	Shared secret (the same one that is used in eIDAS HProxy configurations). Should be redefined by the MS and configured accordingly in the other components using it.
jwt.expiration	600	Expiration time of the jwt token (in seconds) [600 seconds= 10 minutes].
redirect.to.eidasHProxy	http://<hostname>:<port>/eidas-hproxy/eidas-hproxy/authentication	eIDAS HProxy endpoint.
ncphproxy.url	http://<hostname>:<port>/healthied-ncp-hproxy/ncphproxy	NCP HProxy endpoint.

spring.jpa.hibernate.ddl-auto	update	Enables Hibernate to create the ehealth_healtheid schema tables upon deployment of the artefact.
spring.jpa.generate-ddl	true	Switches on/off the JPA DDL generation.
hibernate.dialect	org.hibernate.dialect.MySQL5Dialect	Database SQL Dialect. May need to be changed according to the specific database (and version) used.
spring.datasource.jndi-name	java:comp/env/jdbc/ConfHeiD	Externalizes JNDI configuration declared in tomcat.
acknowledge.path	classpath:acknowledge/default.html	Classpath for the acknowledge HTML page.
pin.reference.version	1.0.0.RC1	Acknowledged PIN-B version. It must be redefined by the MS, according to the version of the PIN-B they're using.
name.sp	<Country-B> Service Provider	The name of the country-B Service Provider.

There are additional properties that configure the Notification Adapter. They'll be explained in section 1.7.5.

The following instructions allow customization of the previous configurations.

Add the following configuration in Tomcat's context.xml:

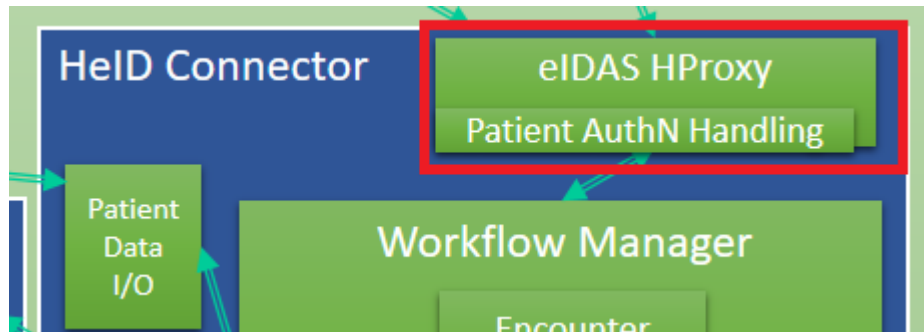
```
<Parameter name="healtheid-connector.properties"
value="/path/to/tomcat/properties/healtheid-connector.properties"/>
```

Where:

- name: it must be `healtheid-connector.properties`;
- value: absolute path to a custom properties file, e.g., can be within a newly created properties file inside Tomcat (but this is not mandatory, it can be anywhere in the filesystem, as long as the user running the HeID Connector has sufficient permissions to read it).

The configurations contained in `healtheid-connector.properties` file will overwrite the default configurations provided by the `default.properties` file included in the component artefact. If this file doesn't exist, the default ones apply.

1.7.2 eIDAS-HProxy



After the patient clicks on the encounter link, he is redirected to eIDAS-HProxy, which has to create an eIDAS SAML Authentication Request towards the eIDAS world. Before doing that, the eIDAS-HProxy receives the JWT token (EncounterID) as a parameter from the WM so that it can read the Citizen Country attribute from this token. Then it builds the eIDAS AuthN request generating a random SAML ID. This SAML ID identifies the eIDAS request, and it is useful to map the corresponding answer. However, the eIDAS HProxy must also map the encounter with the corresponding eIDAS SAML request. For this reason, it uses a HashMap <encounterID, SAML-ID>. When the authentication request is created, the eidas-hproxy adds the entry <encounter-id,saml-id> in the HashMap and sends the request toward eIDAS.

In such way, when the eIDAS Assertion will come, the SAML InResponseTo attribute will contain the SAML-ID; from this value, the system can recover the encounterID stored in the HashMap and link SAML Response, containing the patient attributes, with the encounter, which identifies the patient-HP relationship.

Please note that with this approach, the eIDAS-HProxy has to know the hidden secret wherewith the token is encrypted. Another solution could be to use asymmetric cryptography to sign the JWT token, rather than a shared-secret solution (this requires some changes in the token generation).

The eIDAS HProxy component is based on version 2.2.0 of eIDAS sources in order to use OpenSAML v3, instead of the deprecated v2 of eIDAS 1.4.3. Even though the component is bundled as a separate component, this allow its coexistence with other components using the same version of this library. The sub-components of eIDAS HProxy are based on version 1.4.3 of eIDAS, thus compatible and in line with the configurations from such version.

Please note that eIDAS-HProxy needs to read the JWT Token to discover the Patient Country, so the property described below for the JWT Token secret must be the same that was put in the HeID-connector configuration file.

The eIDAS HProxy component is packaged as a deployable WAR file (eidas-hproxy). For testing purposes, it can be launched as a standalone Spring Boot JAR file containing an embedded Tomcat.

- Prerequisites

- 1) Java 1.8 and Maven
- 2) Firewall configured in order to accept incoming connections on ports 80 (HTTP) and 443 (HTTPS)

- Installation

- 1) Download and extract EIDAS-Sources v2.2.0 from <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+version+2.2>
- 2) Change directory to EIDAS-Parent inside (after extracting EIDAS-Sources-2.2.0.zip)
- 3) Execute the following command in order to install the necessary eu.eidas:eidas-saml-engine artifacts in the local repository.
mvn clean install -DskipTests
- 4) Change directory to eidas-hproxy (inside the health-eid project directory)
git clone https://ec.europa.eu/cefdigital/code/scm/ehncp/health-eid.git
git checkout develop
- 5) Run **mvn clean install -DskipTests** in order to build the application
- 6) The generated WAR file will be located in the eidas-hproxy/target directory for deployment to tomcat.
- 7) It is recommended to setup tomcat to use an SSL/TLS certificate

- Configuration

This component's behavior is managed by a set of properties within its self-contained default.properties file. The following instructions allow customization of such configurations.

Add the following configuration in Tomcat's context.xml (the file is located in the conf folder of the local Tomcat installation):

```
<Parameter name="eidas-hproxy.properties" value="/path/to/tomcat/properties/eidas-hproxy.properties"/>
```

Where:

- name: it must be eidas-hproxy.properties;

- value: absolute path to a custom properties file, e.g., can be within a newly created properties file inside Tomcat (but this is not mandatory, it can be anywhere in the filesystem, as long as the user running the eIDAS HProxy has sufficient permissions to read it).

The configurations contained in eidas-hproxy.properties file will override the default configurations provided by the default.properties file included in the component artifact. If this file doesn't exist, the default ones apply. The following table provides information on the configuration properties:

Name	Value	Description
server.port	8084	TCP port on which the service is run (only used for testing when run as a standalone application with embedded tomcat).
jwt.secret	shared-secret-to-sign-and-verify-JWT-token-change-it	Secret key to verify the JWT token sent by workflow manager. (The same one that is used in HeID Connector configuration)
url.heidConnector.acceptPatientAuth	http://[ip]:[port]/healthid-connector/heidconnector/acceptPatientAuthN	URL to redirect eIDAS result towards workflow manager (check port in healthid-connector conf file)
eidas.path	[PATH]	File system path pointing to eIDAS configuration folder (containing Signing and Encryption settings and keystore). Note: needs to end with a slash (/).
sp.metadata.url	http://[ip]:[port]/eidas-hproxy/eidas-hproxy/metadata	The local endpoint metadata URL as it is visible from the Internet.
sp.country	CA	The code of the country in which the component is deployed

country.metadata.url	http://[ip]:[port]/ConnectorResponderMetadata	eIDAS Connector (or national adapter) metadata URL
sp.return	http://[ip]:[port]/eidas-hproxy/eidas-hproxy/AuthResponse	The local endpoint (SP) return page URL as it is visible from the Internet.
country.nameid.format	urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified	<p>This sets the <saml2p:NameIDPolicy Format> entry in the SAML Authentication request message. It is used to specify the format in which the NameID should be received in the Authn Response message.</p> <p>Change the value if required by the eIDAS Connector. Other possible values are:</p> <ul style="list-style-type: none"> • urn:oasis:names:tc:SAML:2.0:nameid-format:persistent • urn:oasis:names:tc:SAML:2.0:nameid-format:transient • urn:oasis:names:tc:SAML:1.1:nameid-format:persistent
provider.name sp.type contact.support.email contact.support.company contact.support.givenname contact.support.surname	DEMO-SP Public contact.support@sp.eu eIDAS SP Operator Jean-Michel Folon	SP and contact information to be displayed on the metadata Page

<p>contact.support.phone contact.technical.email contact.technical.company contact.technical.givenname contact.technical.surname contact.technical.phone</p>	<p>+555 123456 contact.support@sp.eu eIDAS SP Operator Alphonse Michaux +555 123456</p>	
<p>sp.metadata.retention</p>	<p>86400</p>	<p>The eIDAS connector should re-load the SP metadata page if the following time (in seconds) has elapsed</p>
<p>encryption.algorithm.whitelist</p>	<p>http://www.w3.org/2009/xmlenc11#aes128-gcm;http://www.w3.org/2009/xmlenc11#aes256-gcm;http://www.w3.org/2009/xmlenc11#aes192-gcm</p>	<p>Contains the encryption algorithms allowed in the responses received.</p>
<p>signature.algorithm.whitelist</p>	<p>http://www.w3.org/2001/04/xmldsig-more#rsa-sha256;http://www.w3.org/2001/04/xmldsig-more#rsa-sha384;http://www.w3.org/2001/04/xmldsig-more#rsa-sha512;http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160;http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256;http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384;http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512;http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1</p>	<p>It contains OpenSAML's supported signing algorithms, separated by ";".</p>
<p>eidas.protocol.version</p>	<p>1.1</p>	<p>Value of eIDAS protocol version followed by the SP. When not empty, the value will be published in the SP's metadata URL.</p>
<p>eidas.application.identifier</p>	<p>CEF:eIDAS-ref:1.4.3</p>	<p>Value of eIDAS protocol application identifier relative to the IdP code and version number.</p>

		When not empty, the value will be published in the SP's metadata URL.
--	--	---

- eIDAS Specific Configuration

A sample configuration folder is provided along with the component which should be configured according to the requirements of the eIDAS Connector (or National Adapter component). The folder structure contains the keystore for certificates, as well as configuration files in xml format which are described below.

Folder structure

eidas-demo-config/keystore/	This folder contains the java keystore(s) (in JKS or P12 format) which will contain the certificate(s). The default password is "local-demo".
eidas-demo-config/server/sp/	XML files for configuring Encryption, Signing and other aspects of eIDAS

XML file description

EncryptModule_SP.xml	This file is used to select the keystore and certificate which will be used to decrypt the SAML message sent from the eIDAS connector (or national adaptor)
saml-engine-additional-attributes.xml	Defines eIDAS Additional attributes. It is not necessary to make any changes to this file
saml-engine-eidas-attributes.xml	Defines eIDAS attributes. It is not necessary to make any changes to this file
SamlEngine_SP.xml	SAML constants for Authentication Requests and Responses. It is not necessary to make any changes to this file
SignModule_SP.xml	This file is used to select the keystore and certificate which will be used to sign metadata and SAML messages

SPSamlEngine.xml	This is the root configuration file which links the rest of the files. It is not necessary to make any changes to this file
------------------	---

New Keystore Setup

- 1) Create the file named openssl.cnf:

```
[ req ]
distinguished_name=req_distinguished_name

[ req_distinguished_name ]

[ eidas_sign ]
basicConstraints=critical,CA:FALSE
keyUsage=digitalSignature,nonRepudiation

[ eidas_enc ]
basicConstraints=critical,CA:FALSE
keyUsage=keyEncipherment
```

- 2) Run the following commands to create the X.509 certificates required to sign SAML requests and responses

```
$ openssl ecparam -genkey -name secp384r1 -out priv/eidas-hproxy-sign.key
$ openssl req -x509 -new -key priv/eidas-hproxy-sign.key -sha512 \
  -out pub/eidas-hproxy-sign.pem -days 3650 \
  -config openssl.cnf -extensions eidas_sign \
  -subj "/C=IT/O=Politecnico di Torino/OU=HealtheID Test Infrastructure/CN=eIDAS-HProxy SAML Signature"
```

to encrypt SAML responses

```
$ openssl req -x509 -nodes -newkey rsa:4096 -keyout priv/eidas-hproxy-enc.key -sha512 \
  -out pub/eidas-hproxy-enc.pem -days 3650 \
  -config openssl.cnf -extensions eidas_enc \
  -subj "/C=IT/O=Politecnico di Torino/OU=HealtheID Test Infrastructure/CN=eIDAS-HProxy SAML Encryption"
```

and to sign SAML metadata

```
$ openssl ecparam -genkey -name secp384r1 -out priv/eidas-hproxy-meta.key
$ openssl req -x509 -new -key priv/eidas-hproxy-meta.key -sha512 \
  -out pub/eidas-hproxy-meta.pem -days 3650 \
  -config openssl.cnf -extensions eidas_sign \
  -subj "/C=IT/O=Politecnico di Torino/OU=HealtheID Test Infrastructure/CN=eIDAS-HProxy SAML Metadata Signature"
```

You can check the content of the X.509 certificates by running the command

```
$ openssl x509 -in /path/to/certificate.pem -text -noout
```


Note: the algorithm used to generate the key for the signature of SAML requests/responses and metadata must be of the same family of one of the algorithms officially supported by eIDAS:

- eIDAS supported signature methods requiring an RSA key:
 - <http://www.w3.org/2001/04/xmldsig-more#rsa-sha256>
 - <http://www.w3.org/2001/04/xmldsig-more#rsa-sha384>
 - <http://www.w3.org/2001/04/xmldsig-more#rsa-sha512>
 - <http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160>
 - <http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1>
- eIDAS supported signature methods requiring an EC key:
 - <http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256>
 - <http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384>
 - <http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512>

And the signature method algorithm which belongs to the family of the chosen key generation algorithm must be declared in the signature.algorithm entry of the SignModule_SP.xml (see Module Configuration section). E.g., in the instructions provided previously we used the secp384r1 algorithm for generating the signature key with openssl, which is of the same family as the <http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512>, thus the latter must be configured as the signature.algorithm entry of the SignModule_SP.xml.

- 3) Generate a password for the keystore (e.g. using the pwgen tool) and put the password file (named eidas-hproxy.jks.pwd) in the jks sub-directory

```
# apt-get install pwgen
$ echo -n `pwgen -s 16 1` > jks/eidas-hproxy.jks.pwd
```

- 4) Export the keys and certificates to PKCS-12 archives

```
$ openssl pkcs12 -export -in pub/eidas-hproxy-sign.pem \
-inkey priv/eidas-hproxy-sign.key \
-certfile pub/eidas-hproxy-sign.pem -out p12/eidas-hproxy-sign.p12 \
-name "eidas-hproxy-sign" \
-password pass:`cat jks/eidas-hproxy.jks.pwd`

$ openssl pkcs12 -export -in pub/eidas-hproxy-enc.pem \
-inkey priv/eidas-hproxy-enc.key \
-certfile pub/eidas-hproxy-enc.pem -out p12/eidas-hproxy-enc.p12 \
-name "eidas-hproxy-enc" \
-password pass:`cat jks/eidas-hproxy.jks.pwd`

$ openssl pkcs12 -export -in pub/eidas-hproxy-meta.pem \
```

```
-inkey priv/eidas-hproxy-meta.key \  
-certfile pub/eidas-hproxy-meta.pem -out p12/eidas-hproxy-meta.p12 \  
-name "eidas-hproxy-meta" \  
-password pass:`cat jks/eidas-hproxy.jks.pwd`
```

5) Convert the keys and certificates in the PKCS-12 archive to a new Java KeyStore

```
$ keytool -importkeystore -destkeystore jks/eidas-hproxy.jks \  
-srckeystore p12/eidas-hproxy-sign.p12 -srcstoretype pkcs12 \  
-alias eidas-hproxy-sign -destkeypass `cat jks/eidas-hproxy.jks.pwd` \  
-deststorepass `cat jks/eidas-hproxy.jks.pwd` \  
-deststoretype jks -destalias eidas-hproxy-sign \  
-srcstorepass `cat jks/eidas-hproxy.jks.pwd`  
  
$ keytool -importkeystore -destkeystore jks/eidas-hproxy.jks \  
-srckeystore p12/eidas-hproxy-enc.p12 -srcstoretype pkcs12 \  
-alias eidas-hproxy-enc -destkeypass `cat jks/eidas-hproxy.jks.pwd` \  
-deststorepass `cat jks/eidas-hproxy.jks.pwd` \  
-deststoretype jks -destalias eidas-hproxy-enc \  
-srcstorepass `cat jks/eidas-hproxy.jks.pwd`  
  
$ keytool -importkeystore -destkeystore jks/eidas-hproxy.jks \  
-srckeystore p12/eidas-hproxy-meta.p12 -srcstoretype pkcs12 \  
-alias eidas-hproxy-meta -destkeypass `cat jks/eidas-hproxy.jks.pwd` \  
-deststorepass `cat jks/eidas-hproxy.jks.pwd` \  
-deststoretype jks -destalias eidas-hproxy-meta \  
-srcstorepass `cat jks/eidas-hproxy.jks.pwd`
```

- Module configuration

1) Replace the SignModuleSP.xml file with the following content

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">  
  
<properties>  
  <comment>SWModule sign with JKS.</comment>  
  <entry key="check_certificate_validity_period">true</entry>  
  <entry key="disallow_self_signed_certificate">>false</entry>  
  <entry key="signature.algorithm">http://www.w3.org/2001/04/xmldsig-more#ecdsa  
-sha512</entry>  
  <entry key="signature.algorithm.whitelist">  
    http://www.w3.org/2001/04/xmldsig-more#rsa-sha256;  
    http://www.w3.org/2001/04/xmldsig-more#rsa-sha384;  
    http://www.w3.org/2001/04/xmldsig-more#rsa-sha512;  
    http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160;  
    http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256;  
    http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384;  
    http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512  
  </entry>  
  <entry key="response.sign.assertions">true</entry>  
  <entry key="keyStorePath">${KEYSTORE}</entry>  
  <entry key="keyStorePassword">${PASSWORD}</entry>  
  <entry key="keyPassword">${PASSWORD}</entry>  
  <entry key="issuer">C=IT, O=Politecnico di Torino, OU=HealtheID Test Infrastr  
ucture, CN=eIDAS-HProxy SAML Signature</entry>  
  <entry key="serialNumber">${SIGN_SERIAL_NUMBER}</entry>  
  <entry key="keyStoreType">JKS</entry>  
  <entry key="metadata.keyStorePath">${KEYSTORE}</entry>  
  <entry key="metadata.keyStorePassword">${PASSWORD}</entry>
```

```
<entry key="metadata.keyPassword">${PASSWORD}</entry>
<entry key="metadata.issuer">C=IT, O=Politecnico di Torino, OU=HealtheID Test
Infrastructure, CN=eIDAS-HProxy SAML Metadata Signature</entry>
<entry key="metadata.serialNumber">${META_SERIAL_NUMBER}</entry>
<entry key="metadata.keyStoreType">JKS</entry>
</properties>
```

where:

- \$KEYSTORE is the full path name of eidas-hproxy.jks
- \$PASSWORD is the password used by eidas-hproxy.jks (content of the eidas-hproxy.jks.pwd file)
- \$SIGN_SERIAL_NUMBER is the serial number of the certificate used to sign SAML request and responses
- \$META_SERIAL_NUMBER is the serial number of the certificate used to sign SAML metadata

2) Replace the EncryptModule_SP.xml file with the following content

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>SWModule encrypt with JKS.</comment>
  <entry key="check_certificate_validity_period">true</entry>
  <entry key="disallow_self_signed_certificate">>false</entry>
  <entry key="response.encryption.mandatory">true</entry>
  <entry key="data.encryption.algorithm">http://www.w3.org/2009/xmlenc11#aes256-gcm</entry>
  <entry key="encryption.algorithm.whitelist">
    http://www.w3.org/2009/xmlenc11#aes128-gcm;
    http://www.w3.org/2009/xmlenc11#aes256-gcm;
    http://www.w3.org/2009/xmlenc11#aes192-gcm
  </entry>
  <entry key="key.encryption.algorithm">http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p</entry>
  <entry key="keyStorePath">${KEYSTORE}</entry>
  <entry key="keyStorePassword">${PASSWORD}</entry>
  <entry key="keyPassword">${PASSWORD}</entry>
  <entry key="keyStoreType">JKS</entry>
  <entry key="encryptionActivation">encryptionConf.xml</entry>
  <entry key="responseDecryptionIssuer">C=IT, O=Politecnico di Torino, OU=HealtheID Test Infrastructure, CN=eIDAS-HProxy SAML Encryption</entry>
  <entry key="serialNumber">${ENC_SERIAL_NUMBER}</entry>
</properties>
```

where:

- \$KEYSTORE is the full path name of eidas-hproxy.jks
- \$PASSWORD is the password used by eidas-hproxy.jks (content of the eidas-hproxy.jks.pwd file)
- \$ENC_SERIAL_NUMBER is the serial number of the certificate used to encrypt attributes contained in SAML responses

The values for issuer and serialNumber can be obtained by listing certificates inside the keystore.

keytool -list -v -keystore eidas-hproxy.jks

Note: When working with self-signed certificates, please make sure that the following entry is set to false. Otherwise, set to true.

```
<entry key="disallow_self_signed_certificate">false</entry>
```

Importing eIDAS Connector (or National Adapter) signing certificate to the keystore

In order for eidas-hproxy component to trust the eIDAS Connector (or National Adapter), it is necessary to add its public certificate in the keystore (which was created in the previous steps – it must be the keystore configured in the metadata.keyStorePath entry of SignModule_SP.xml).

- Navigate to the eIDAS Connector (or National Adapter) metadata URL using a web browser.
- Locate the public certificate used for signing. This is located in section `<md:IDPSSODescriptor>` between the tags `<ds:X509Certificate>` and `</ds:X509Certificate>`.
- Extract the certificate to a file. For this you need to copy paste the certificate to a new text file. Then add “-----BEGIN CERTIFICATE-----” and “-----END CERTIFICATE-----” statements to the beginning and end of the file respectively. The file should look like this:

```
-----BEGIN CERTIFICATE-----  
MIICNTCCAZ6gAwIBAgIES343gjANBgkqhkiG9w0BAQUFADBVMQswCQYDVQQGEwJVUzELMAkGA1UE  
CAwCQ0ExFjAUBgNVBAcMDU1vdW50YWluIFZpZxcxDTALBgNVBAoMBBFdTTzlxEjAQBgNVBAMMCWxv  
...  
...  
QIRG5ITCZXy9hi0PygLP2rHANh+PYfTmxbuOnykNGyhM6FjFLbW2uZHqTY1jMrPprjOrmyK5sjJR  
O4d1DeGHT/Ynljs9JogRKv4XHECwLtlVdAbldWHEtVZJyMSkcyysFcvuhPQK8Qc/E/Wq8uHSCo=  
-----END CERTIFICATE-----
```

If the issuer of the certificate is not trusted, you need the full certificate chain.

- Import the newly created certificate file to the keystore using the following command
`keytool -importcert -file connector.crt -keystore eidas.jks`

You will be prompted if you wish to trust the certificate and if successful, the command will output “Certificate was added to keystore”

Example error output when the certificate has not been added correctly:

Caused by: eu.eidas.engine.exceptions.EIDASSAMLEngineException: Error (no.

samlengine.untrusted.certificate.code) processing request :

samlengine.untrusted.certificate.message - null

at

eu.eidas.auth.engine.core.impl.AbstractProtocolSigner.checkValidTrust(AbstractProtocolSigner.java:409)

Other common issues you may encounter

- Although changes are made to the eidas-hproxy component (such as `sp.return url`), these seem to be ignored. This may happen because the eIDAS Connector (or National Adapter) has stored the eidas-hproxy metadata in cache. The metadata is stored for a duration configured in `sp.metadata.retention` (default is 24h) and this is expected behavior. Please allow the time to expire, or use a smaller retention value during the initial setup.

- b) During download of the remote metadata, the following error appears:
`javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException:
PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
The SSL/TLS (not eIDAS/SAML) certificate of the eIDAS Connector (or
National Adapter) is not trusted by the local Java installation. You will need
to export the public certificate using a browser, determine the location of
cacerts (Java CA certificates store) and import the certificate using the
following command:
keytool -import -alias example -keystore [local-path]/cacerts -file example.cer`
- c) In case your eIDAS Connector (or National Adapter) is exposing a metadata
signed with an algorithm different than the ones listed in the
signature.algorithm.whitelist entry of the SignModule_SP.xml file (e.g., RSA-
SHA1), you'll need to edit the SAML engine project, namely the class
eu/eidas/auth/engine/core/impl/AbstractProtocolSigner.java and add the
specific algorithm to all the ImmutableSets (i.e., Signature and Digest), and
then generate a new version of the eidas-saml-engine JAR file to be included
as a dependency of the eIDAS-HProxy. Plus, it's also needed to add that
algorithm to the signature.algorithm.whitelist entry of the
SignModule_SP.xml and of the eidas-hproxy.properties.
- d) In case you see an error similar to:
- o Caused by:
eu.eidas.auth.engine.configuration.ProtocolEngineConfigurationExce
ption: Error (no. null) processing request : No private key entry
matching serialNumber=XXXXXX and issuer=CN=something.gov,
OU=MyOU, O=MyO, L=MyL, C=MyC, EMAILADDRESS=my-
email@gov.com found in configured keyStore - null

This might be due to a slight difference in the RFC representation of the
certificates (experience with some certificates showed a single character
differing, with no clear root cause to this).

You might need to change the eIDAS Encryption project, namely the
eu.eidas.auth.engine.xml.opensaml.CertificateUtil.matchesCertificate
method in the following way and regenerate the eidas-encryption JAR file
which is a dependency of eIDAS-HProxy:

```
public static boolean matchesCertificate(String serialNumber, String issuer,
X509Certificate certificate) {
    if (null == certificate) {
        return false;
    }
    BigInteger serialNumberBigInteger = new BigInteger(serialNumber, 16);
    BigInteger certificateSerialNumber = certificate.getSerialNumber();
```

```
X500Principal issuerPrincipal = new X500Principal(issuer);
X500Principal certificateSubjectPrincipal =
certificate.getSubjectX500Principal();
//create the X500Principal based on the string representation of the X.500
distinguished name using the format defined in RFC 2253
X500Principal unencodedCertificateSubjectPrincipal = new
X500Principal(certificateSubjectPrincipal.getName());
X500Principal certificateIssuerPrincipal =
certificate.getIssuerX500Principal();
//create the X500Principal based on the string representation of the X.500
distinguished name using the format defined in RFC 2253
X500Principal unencodedCertificateIssuerPrincipal = new
X500Principal(certificateIssuerPrincipal.getName());
String strprincipalKeystore = certificate.getIssuerDN().toString();
X500Principal issuerPrincipalKeystore = new
X500Principal(strprincipalKeystore);
return serialNumberBigInteger.equals(certificateSerialNumber) && (
    issuerPrincipal.equals(unencodedCertificateSubjectPrincipal) ||
    issuerPrincipal.equals(
        issuerPrincipalKeystore));
}
```

Metadata generation verification

Load <https://<hostname/ip>:<port>/eidas-hproxy/eidas-hproxy/metadata> from a web browser and examine the resulting XML metadata page. Verify the following information:

- entityID should match the sp.metadata.url e.g.

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://heid-connector.test/eidas-hproxy/metadata" validUntil="2019-09-25T19:30:09.882Z">
```

- The metadata should be signed e.g.

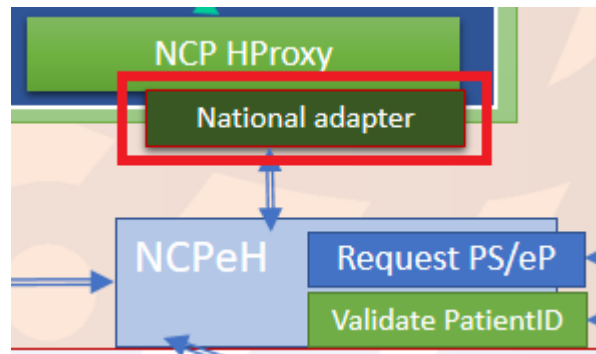
```
<ds:SignatureValue>...</ds:SignatureValue>
```

- Verify that the public X509 Certificates appear
- Verify that the return page is set correctly

```
<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="http://heid-connector.test/eidas-hproxy/AuthResponse" index="0" isDefault="true"/>
```

```
<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="http://heid-connector.test/eidas-hproxy/AuthResponse" index="1"/>
```

1.7.3 NCP National Adapter



This National Adapter (to not confuse with the eIDAS National Adapter) allows the connection between the NCP HProxy of the HeID Connector and the NCPeH-B, requesting from the latter the country-A configuration (international search mask and eIDAS eHealth configuration). It consists of two projects:

- **openncp-national-adapter-interface**: this project will be inside the healtheid-ncp-hproxy project as a JAR file. It provides the interface towards which the NCP HProxy sends requests.
- **openncp-national-adapter**: this project will be inside the healtheid-ncp-hproxy project as a JAR file. It is a default implementation of the National Adapter, provided by HEALTHeID, ready to connect to a default implementation/deployment of the OpenNCP, through the HEALTHeID-enhanced OpenNCP CC Web Services Client Consumer component (openncp-ncp-pt-client-consumer).

Should a MS wish to use its own National Adapter, for the purposes of connecting to a nationally customized NCPeH-B, the NCP HProxy (healtheid-ncp-hproxy) component defines a Maven profile – “national-build” – that gives the MS such flexibility, pretty much in the same way they already do in eHDSI with the National Connector of NCPeH-A.

```

<dependency>
  <groupId>${healtheid.openncp.national-adapter.groupId}</groupId>
  <artifactId>${healtheid.openncp.national-adapter.artifactId}</artifactId>
  <version>${healtheid.openncp.national-adapter.version}</version>
</dependency>
  
```

The National Adapter dependency is parameterized and can be defined within the pom.xml of the healtheid-ncp-hproxy Maven project.


```
<profiles>
  <profile>
    <id>national-build</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
      <healtheid.openncp.national-adapter.groupId>eu.europa.ec.healtheid</healtheid.openncp.national-adapter.groupId>
      <healtheid.openncp.national-adapter.artifactId>openncp-national-adapter</healtheid.openncp.national-adapter.artifactId>
      <healtheid.openncp.national-adapter.version>3.0.0.RC3</healtheid.openncp.national-adapter.version>
    </properties>
  </profile>
</profiles>
```

It must be highlighted that, even in the case of a MS custom National Adapter, the HEALTHeID-enhanced OpenNCP CC Web Services Client Consumer component (openncp-ncp-pt-client-consumer) must be used further down the road, in order to properly communicate with the HEALTHeID version of the OpenNCP Client Connector (openncp-client-connector).

Figure 3 illustrates how the components can handle two different implementations of the National Adapter.

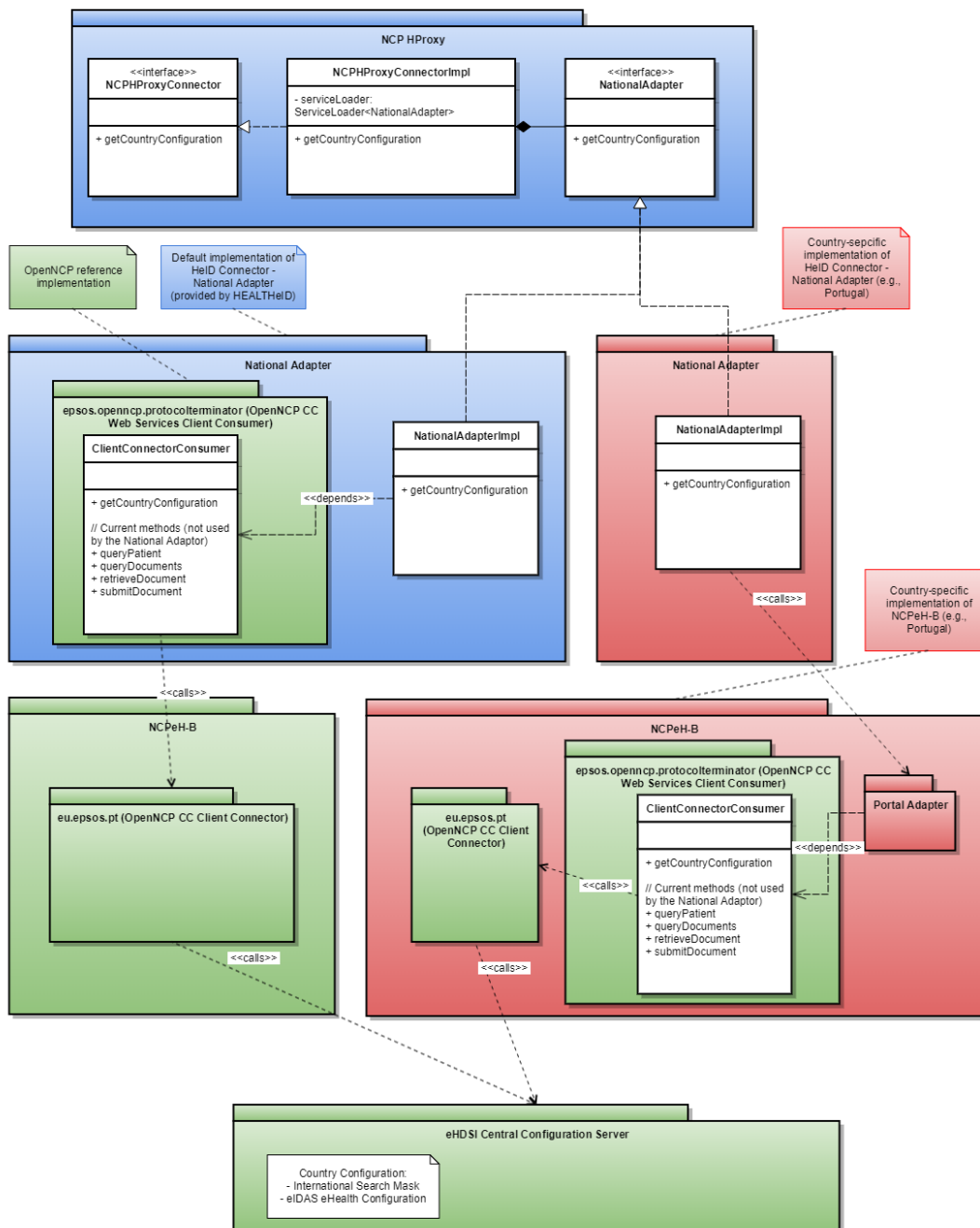
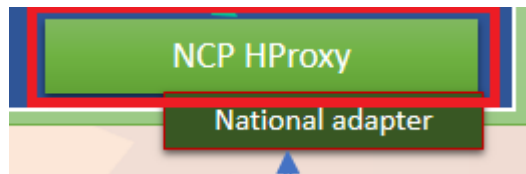


Figure 3 - National Adaptor integration in HEALTHeID

1.7.4 NCP HProxy



The NCP HProxy component is packaged as a deployable WAR file (healtheid-ncp-hproxy). For testing purposes, it can be launched as a standalone Spring Boot JAR file containing an embedded Tomcat.

This component's behavior is managed by a set of properties within its self-contained default.properties file. The following instructions allow customization of such configurations.

Add the following configuration in Tomcat's context.xml:

```
<Parameter name="ncp-hproxy.properties"
value="/path/to/tomcat/properties/ncp-hproxy.properties"/>
```

Where:

- name: it must be ncp-hproxy.properties;
- value: absolute path to a custom properties file, e.g., can be within a newly created properties file inside Tomcat (but this is not mandatory, it can be anywhere in the filesystem, as long as the user running the NCP HProxy has sufficient permissions to read it).

The configurations contained in ncp-hproxy.properties file will overwrite the default configurations provided by the default.properties file included in the component artefact. If this file doesn't exist, the default ones apply. The following table provides information on the configuration properties:

Name	Value	Description
jwt.secret	shared-secret-to-sign-and-verify-JWT-token-change-it	JWT secret. Should be redefined by the MS. Currently not being validated.
jwt.expiration	7200	JWT Expiration time (in seconds). Currently not being validated.
server.ssl.enabled	true	Enable HTTPS. If used, it should be "true". Only used in the case the component is run as a standalone JAR file. Otherwise it can be

		ignored, given that such configuration is provided by the application server where it's deployed.
security.require-ssl	true	Requires SSL. If used, it should be "true". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store-type	JKS	The format used for the keystore. It could be set to JKS in case it is a JKS file. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store	/home/user/health- eid/health- connector/keystore/keyst ore.jks	The path to the keystore containing the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.ssl.key-store- password	password	The password used to generate the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided

		by the application server where it's deployed.
server.ssl.key-alias	healtheid-hproxy	The alias mapped to the certificate. Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.tomcat.remote-ip-header	x-forwarded-for	Name of HTTP header from which the remote IP is extracted. If used, it must be "x-forwarded-for". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.
server.tomcat.protocol-header	x-forwarded-proto	Enable setting the header of incoming protocol. If used, it must be "x-forwarded-proto". Only used in the case the component is run as a standalone JAR file. Otherwise it can be ignored, given that such configuration is provided by the application server where it's deployed.

Even though the NCP HProxy is configured to use Spring Security to increase security, at the moment, its only endpoint is freely exposed (i.e., no JWT or any other kind of token validation is performed, as described previously in the `jwt.*` properties). But the component is prepared to be configured accordingly.

Following the previous alternatives for the National Adapter, its consequences on the NCP HProxy deployment are the following:

NCP HProxy default implementation

With the default implementation provided by HEALTHeID, the NCP HProxy must be deployed in the NCP infrastructure, since the default National Adapter depends on the EPSOS_PROPS_PATH environment variable (deeply tied to the OpenNCP reference implementation components – in Figure 3, the OpenNCP CC Web Services Client Consumer). This variable must be available (e.g., via the Tomcat's /bin/setenv.sh file). Additionally, the JNDI resource jdbc/ConfMgr, demanded by the OpenNCP components, must be configured in the Tomcat where the NCP HProxy is deployed (in /conf/context.xml and /conf/server.xml).

Tomcat context.xml:

```
<Context>
  <!-- Default set of monitored resources. If one of these changes, the -->
  <!-- web application will be reloaded. -->
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

  <!-- Uncomment this to disable session persistence across Tomcat restarts -->
  <!--
  <Manager pathname="" />
  -->

  <ResourceLink global="jdbc/ConfMgr" name="jdbc/ConfMgr" type="javax.sql.DataSource"/>
</Context>
```

Tomcat server.xml:

```
<GlobalNamingResources>
  <!-- Editable user database that can also be used by
  UserDatabaseRealm to authenticate users
  -->
  <Resource name="UserDatabase" auth="Container"
    type="org.apache.catalina.UserDatabase"
    description="User database that can be updated and saved"
    factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
    pathname="conf/tomcat-users.xml" />
  <Resource name="jdbc/ConfMgr" auth="Container" factory="com.zaxxer.hikari.HikariJNDIFactory" type="javax.sql.DataSource" singleton="true"
    minimumIdle="2" maximumPoolSize="5" connectionTimeout="300000" dataSourceClassName="com.mysql.jdbc.jdbc2.optional.MysqlDataSource"
    dataSource_serverName="server"
    dataSource_port="3306"
    dataSource_databaseName="database"
    dataSource_user="User"
    dataSource_password="password"/>
</GlobalNamingResources>
```

For the previous configuration, a JDBC Connection Pool such as HikariCP must be available. This is achieved by placing the HikariCP-2.6.3.jar, slf4j-api-1.7.25.jar and mysql-connector-java.jar (e.g., version 5.1.48) in the Tomcat's /lib folder (same versions used by the OpenNCP are used here for ease of installation). Following are the Maven artefacts declaration to help their identification in the Maven Central repository.

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.48</version>
</dependency>
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>3.1.0</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>1.7.25</version>
</dependency>
```

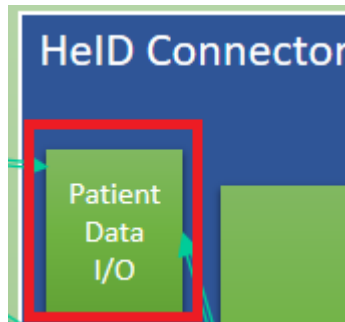
In case another database provider is used, a different JAR file than the MySQL one must be used and the JNDI resource must be configured accordingly.

This default implementation of the NCP HProxy looks for the property `PORTAL_CLIENT_CONNECTOR_URL` of the OpenNCP properties schema (`ehealth_properties`), which should point to the OpenNCP Client Connector deployed at NCP-B (as in a typical OpenNCP installation).

NCP HProxy national implementation

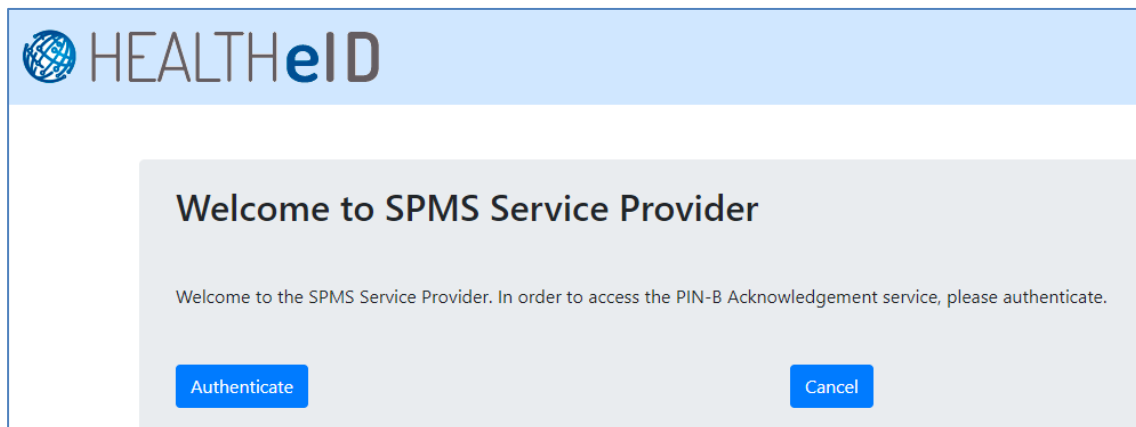
With a national implementation, it may be deployed separately from the NCP infrastructure.

1.7.5 Patient I/O

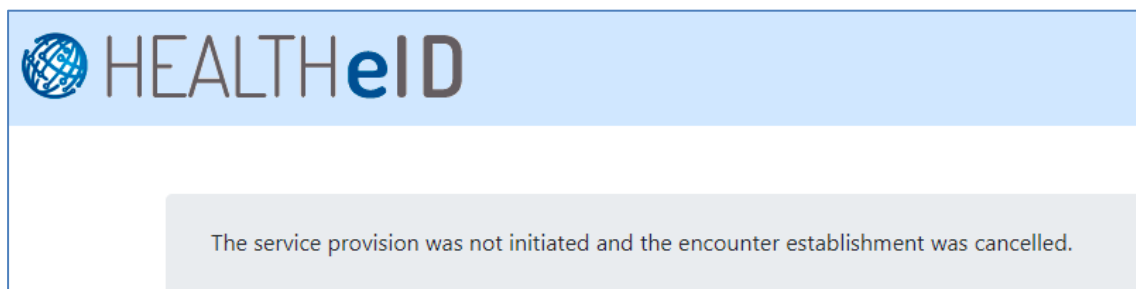


This component is using bootstrap4 (<https://getbootstrap.com/>) as base for CSS.
The following pages will be displayed to the patient:

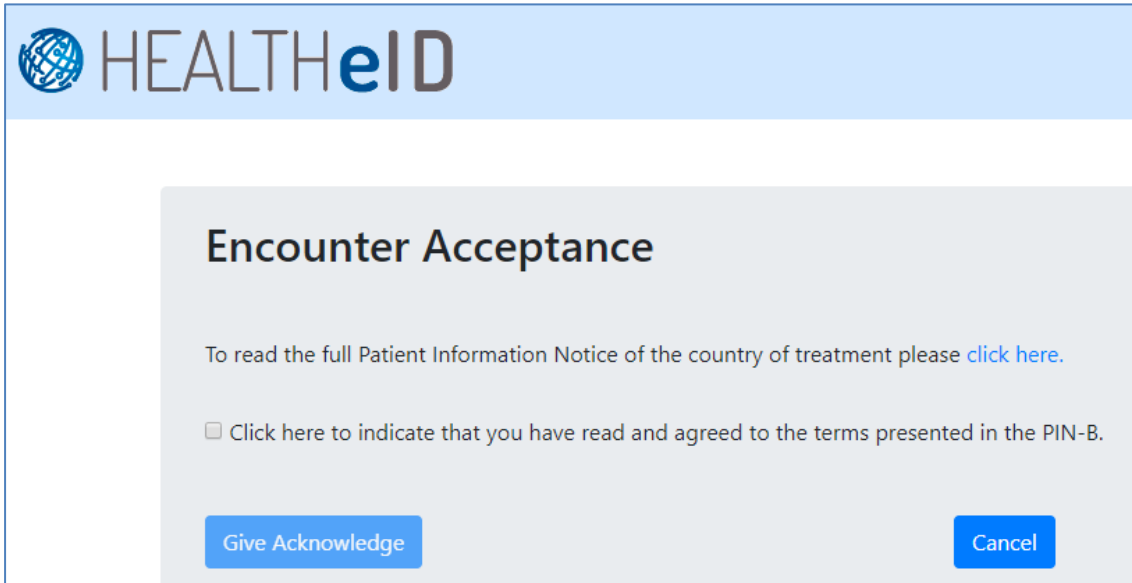
- Welcome page



- Cancel Encounter



- Acknowledge page (“/patientEncounter/patientAcknowledge”): accept acknowledge, dummy model created pointing to project classpath file (acknowledge.path=classpath:acknowledge/default.html) and can be configured to use another file from the source.



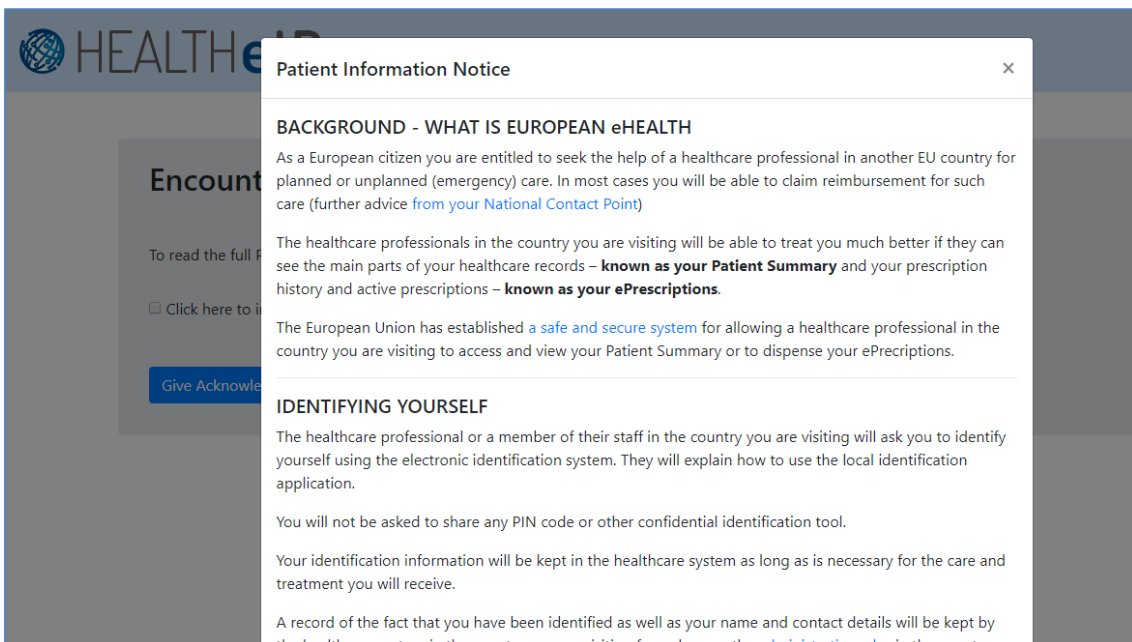
HEALTHeID

Encounter Acceptance

To read the full Patient Information Notice of the country of treatment please [click here](#).

Click here to indicate that you have read and agreed to the terms presented in the PIN-B.

[Give Acknowledge](#) [Cancel](#)



Patient Information Notice [X]

BACKGROUND - WHAT IS EUROPEAN eHEALTH

As a European citizen you are entitled to seek the help of a healthcare professional in another EU country for planned or unplanned (emergency) care. In most cases you will be able to claim reimbursement for such care (further advice [from your National Contact Point](#))

The healthcare professionals in the country you are visiting will be able to treat you much better if they can see the main parts of your healthcare records – **known as your Patient Summary** and your prescription history and active prescriptions – **known as your ePrescriptions**.

The European Union has established a [safe and secure system](#) for allowing a healthcare professional in the country you are visiting to access and view your Patient Summary or to dispense your ePrescriptions.

IDENTIFYING YOURSELF

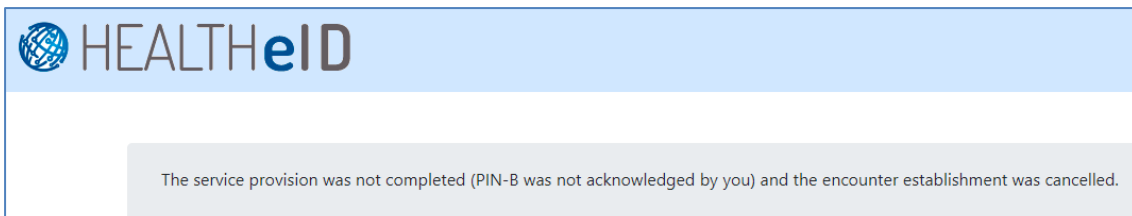
The healthcare professional or a member of their staff in the country you are visiting will ask you to identify yourself using the electronic identification system. They will explain how to use the local identification application.

You will not be asked to share any PIN code or other confidential identification tool.

Your identification information will be kept in the healthcare system as long as is necessary for the care and treatment you will receive.

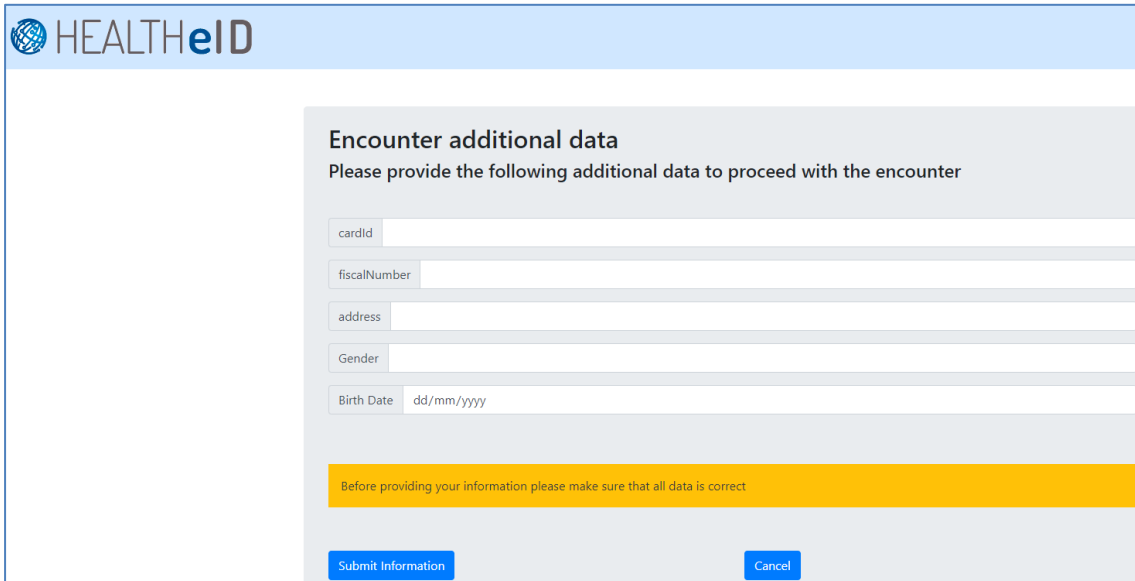
A record of the fact that you have been identified as well as your name and contact details will be kept by the healthcare system in the country you are visiting for as long as the administrative rules in the country...

- **Cancel Encounter**



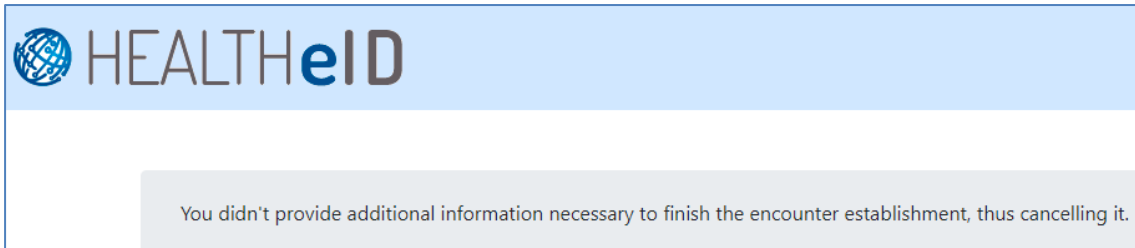
The service provision was not completed (PIN-B was not acknowledged by you) and the encounter establishment was cancelled.

- **Additional Data page (“/patientEncounter/additionalPatientData”):** here the patient adds all the information needed but not provided by eIDAS. This will follow the search mask attributes for mapping purpose.



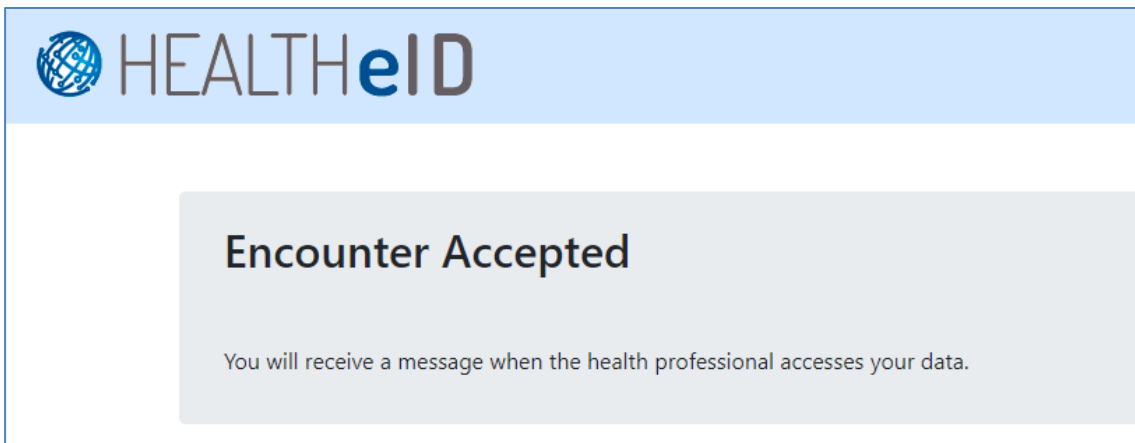
The screenshot shows the 'Encounter additional data' form in the HEALTHeID interface. The form title is 'Encounter additional data' and the instruction is 'Please provide the following additional data to proceed with the encounter'. The form contains several input fields: 'cardId', 'fiscalNumber', 'address', 'Gender', and 'Birth Date' (with a placeholder 'dd/mm/yyyy'). Below the fields is a yellow warning box that says 'Before providing your information please make sure that all data is correct'. At the bottom of the form are two buttons: 'Submit Information' and 'Cancel'.

- Cancel Encounter



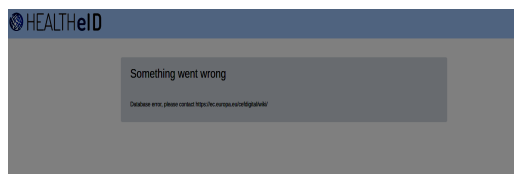
The screenshot shows a message box in the HEALTHeID interface. The message text is 'You didn't provide additional information necessary to finish the encounter establishment, thus cancelling it.'

- Encounter Accepted



The screenshot shows a message box in the HEALTHeID interface. The message title is 'Encounter Accepted' and the message text is 'You will receive a message when the health professional accesses your data.'

- Error Page: This page will be shown if any internal error occur



Notification Adapter

The Notification Adapter component is packaged as a deployable JAR file (healtheid-notification-adapter).

This component implements an email service for Patient notification. It receives the patient email from the Workflow Manager with the kind of notification to send, and sends an email based on the type of notification, e.g., communicates the encounter link to the patient or informs him about what is happening in the OpenNCP world (PS/eP/eD retrieval/submission, PIN Acknowledgement/Consent decisions). The Notification Adapter implementation is required, and each Member State can create its own. This one provides the email feature for demonstration purposes. The application.properties file sets the mail configuration properties to realize the service with Spring. The following table provides information on the configuration properties:

Name	Value	Description
spring.mail.host	smtp.gmail.com	Email server host. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.port	587	Email server port. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.username	testconnector2019@gmail.com	Email authentication username. It should be redefined by the MS

		according to their email infrastructure configuration.
spring.mail.password	TestConnector2019	Email authentication password. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.properties. mail.smtp.starttls.enable	true	Enable StartTLS. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.properties. mail.smtp.starttls.required	true	Require StartTLS. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.properties. mail.smtp.ssl.enable	false	Enables SSL. It should be redefined by the MS according to their email infrastructure configuration.
spring.mail.properties. mail.smtp.auth	true	Enables email authentication. It should be redefined by the MS according to their email infrastructure configuration.

spring.mail.properties. mail.smtp.connectiontime out	5000	Connection timeout (in ms).
spring.mail.properties. mail.smtp.timeout	5000	Timeout (in ms).
spring.mail.properties. mail.smtp.writetimeout	5000	Write timeout (in ms).
email.from	no-reply@company.com	Email sender.

The properties contained in the Notification Adapter’s application.properties file are not set (i.e., they’re commented) since they’re in fact globally set by the HeID-Connector component (in its default.properties file), which includes this one. But they can be set, should you wish to run the component locally, in an isolated way. To overwrite the Notification Adapter default configurations provided by the HeID-Connector default.properties file included within the latter, you must provide the custom values in the healthaid-connector.properties file deployed within the Tomcat (as explained in section 1.7.1).

The necessary network infrastructure and email configurations must be prepared in advance by the MS (e.g., configuring email relay server; network access between the HEALTHeID-Connector infrastructure and email infrastructure).

To enable the sending SMS feature, the module needs of an SMS gateway (demonstration not provided). Considering that the Notification Adapter interacts directly with the Patient, this module can be supposed as part of the Patient Data I/O.

1.7.6 HeID Connector – Flow Example

Following is a technical overview of the steps performed by the HeID Connector during a typical HeID scenario.

1 - createEncounter (step 1 to step 6) :

1a- Generate JWT

- 1b- Store the encounter in DB
- 1c - Send SMS/Email to the Patient
- 1d - Return JWT

- 2 - acceptEncounter - Triggered by the Patient, on click
 - 2a - Validate Token
 - 2b - Redirect to welcome page

- 3 - acceptEncounter - Triggered by the Patient, on click of Authenticate button
 - 3a - Validate Token
 - 3b - Redirect to Login Screen of eIDAS

- 4 - acceptPatientAuthN – Triggered by eIDAS-HPProxy after Patient AuthN
 - 4a - Validate Token
 - 4b - Receives the eIDAS information with the token retrieved by the
InResponseTo SAML attribute
 - 4c - Redirect to acknowledge screen

- 5 - acknowledgeStore (Sent from acknowledge screen)
 - 5a - Validate Token
 - 5b - Store the acknowledge in DB
 - 5c - Send notice to Patient about acknowledgement
 - 5d - Check if it's needed to collect the consent (*not implemented*)
 - 5dd - [optional] - Redirect to Consent page (*not implemented*)
 - 5e - SMP Country Configuration Search
 - 5f - Store health country configuration in DB
 - 5g - Check if manual additional data is needed or not
 - 5gg - [optional] Redirect to Patient Additional Data screen
 - 5h - Update encounter table the field "DataReady" to value "true"
 - 5i - Redirect to encounter accepted page

- 6 - patientConsent (Redirect consent page) [OPTIONAL] (*not implemented*)
 - 6a - Validate Token
 - 6b - Store consent in DB
 - 6c - Send notice to Patient about consent
 - 6d - SMP Country Configuration Search
 - 6e - Store health country configuration in DB
 - 6f - Check if manual additional data is needed or not

- 6ff - [optional] Redirect to Patient Additional Data screen
- 6g - Update encounter table the field “DataReady” to value “true”
- 6h - Redirect to encounter accepted page

7 - additionalPatientData (Sent from the Patient Additional Data screen)
[OPTIONAL]

- 7a - Validate Token
- 7b - Receive information from screen
- 7c - Store additional Data in DB
- 7d - Update encounter table the field “DataReady” to value “true”
- 7d - Redirect to encounter accepted page

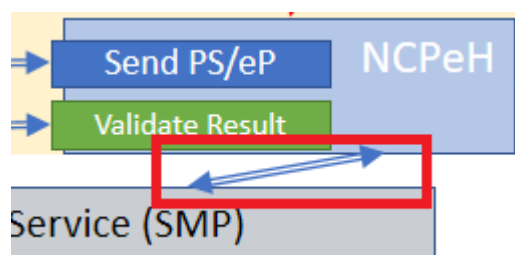
8 - requestPatientData - (Used by HeID Client to retrieve patient authN information,
a poll method is available on openncp-portal)

- 8a - Validate Token
- 8b - Search information saved in the database/memory, using Token as
primary key to search
- 8c - Return all information if it exists

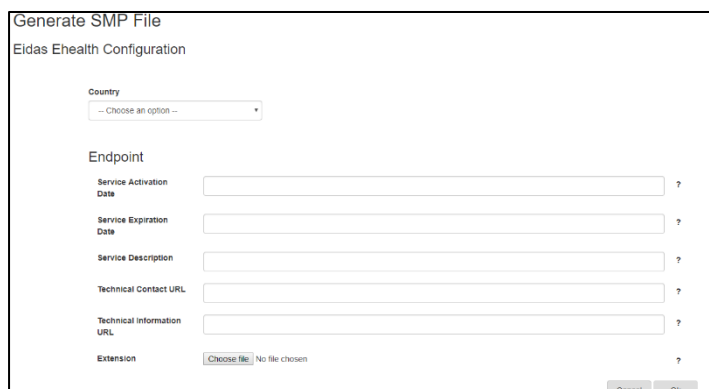
9 – notifyPatient (Will be called twice)

- 9a - Validate Token
- 9b - Notifies Patient that Document or information has been read by HP,
according to parameter passed.
- 9bb – [XCPD] Update patient ID in DB

1.8 NCPeH-A



NCPeH-A is now able to publish in the eHDSI Central Configuration Service (SMP) a new type of SMP file containing configurations related to the application of the eIDAS assertion data in eHealth: eIDAS eHealth Configuration. The OpenNCP-Gateway is the component responsible for the generation and publication of this file.



In the Extension element of this new SMP record, an XML file should be provided. This file should have the following structure (sample values included):

```

<?xml version="1.0" encoding="UTF-8"?>
<EidasEhealthConfiguration xmlns="http://ec.europa.eu/sante/ehncp/eidas">
  <patientInput>
    <patientInputNeeded>true</patientInputNeeded>
    <patientIdMappableEidasAttribute>PersonIdentifier</patientIdMappableEidasAttribute>

    <patientIdMappableEidasAttributeOid>2.16.620.1.101.10.1.3</patientIdMappableEidasAttributeOid>
  </patientInput>
</EidasEhealthConfiguration>
  
```

The schema structure contains 3 elements:

- **patientInputNeeded**: answers to the question “Do we need our patients to input their patientID (and other data)?”;
- **patientIdMappableEidasAttribute**: Friendly name of the eIDAS attribute from which we can derive or map to the patientID;
- **patientIdMappableEidasAttributeOid**: OID of the beforementioned attribute.

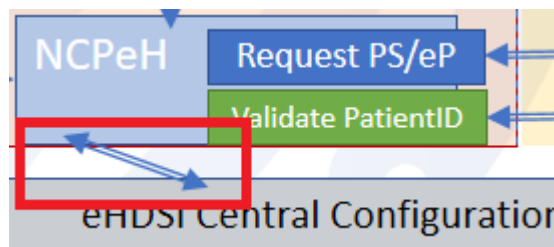
To understand the effects of this configuration in the workflow please consult the following attached spreadsheet:



eidas-assertion-xcp
d-match-20191114.x

To take advantage of HEALTHeID-enhanced OpenNCP-Gateway (openncp-gateway WAR file), the version provided by the HEALTHeID project must be deployed by the country.

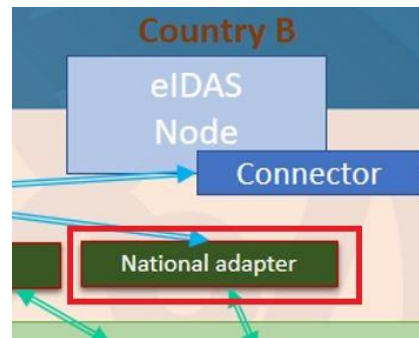
1.9 NCPeH-B



The NCPeH-B side was enhanced with a feature for fetching the eIDAS eHealth Configuration and returning it to the NCP HProxy via the National Adapter, which in turn uses the HEALTHeID-enhanced OpenNCP CC Web Services Client Consumer component, as depicted in Figure 3. This configuration is saved in the NCPeH-B file system, pretty much in the same way already done for the international search masks, in the folder \$EPSOS_PROPS_PATH/eidas, with the filename such as EidasEhealthConfig_CC.xml, where CC is the ISO 3166-1 alpha-2 country code. This folder is created automatically by the component.

To take advantage of these features, the version of the OpenNCP Client Connector (openncp-client-connector WAR file) provided by the HEALTHeID project must be deployed by the country.

1.10 eIDAS National Adapter



The NationalAdapter module has been developed to fulfill the specific Italian needs, when an Italian patient has an encounter with a HP in another Member State, but can be modified in order to be used in other scenarios. It includes two components: NationalAdapter.war and NationalAdapterSpidService.war

NationalAdapter.war module is designed to:

- Validate the eIDAS SAML Request sent by the Service Provider
- Forward the eIDAS SAML Request to the NationalAdapterSpidService component
- Create the eIDAS SAML Response adapting the SPID SAML Response prepared by the NationalAdapterSpidService component
- Send the eIDAS SAML Response to the Service Provider
- Publish online the metadata needed to the Service Provider

NationalAdapterSpidService.war is designed to:

- Create the SPID SAML Request adapting the eIDAS SAML Request
- Send the SPID SAML Request to the Italian eIDAS Proxy
- Validate the SAML Response received from the Italian eIDAS Proxy
- Send the SPID SAML Response to the NationalAdapter module

A Member State willing to use the NationalAdapter component must replace the NationalAdapterSpidService with a specific module focuses on the eIDAS<->national protocol conversion.

It is needed to define the property `national.protocol.converter.url` available in `nationalAdapter.properties` using the URL of the specific national module (which will replace NationalAdapterSpidService).

Furthermore, the servlet `eu.eidas.idp.ProcessResponse` has to be modified in order to work with the national, specific implementation – its goal is to transform the SPID Response sent by the NationalAdapterSpidService into an eIDAS SAML Response and forward it to the Service Provider.

The NationalAdapter module has been developed starting from the eIDAS Node reference implementation, hence it needs some configuration starting from the

original files (i.e. keystore for signature/encryption, white list configuration, etc). To do so, it is recommended to read the installation guide of the eIDAS Node (<https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS-Node+Integration+Package>)

Should a proxy be used to download metadata, all necessary configurations must be defined in the JVM options, as briefly illustrated in the example below:

```
-Dhttps.proxyHost=host  
-Dhttps.proxyPort=port  
-Dhttps.proxyUser=user  
-Dhttps.proxyPassword=password  
-Dhttp.nonProxyHosts=excluded_host
```

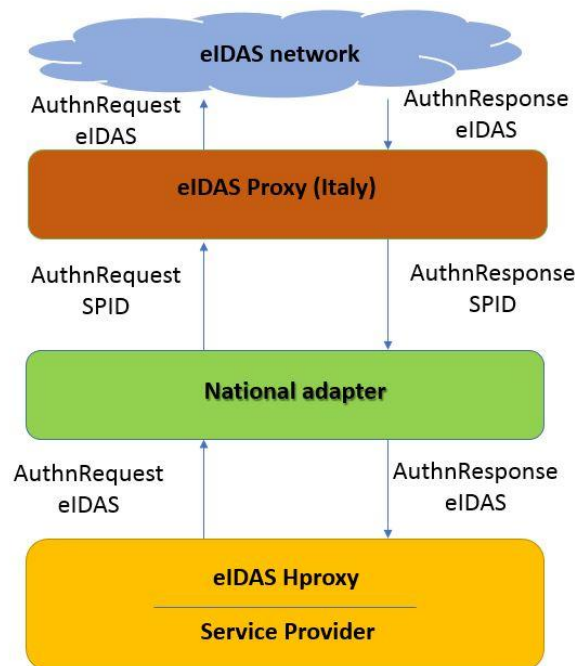
A Member State could be interested just to see the eIDAS authentication flow, without engaging the real operational eIDAS network, e.g. because such a Country could not be ready to be operational in the eIDAS network although interested in the project. It is possible to do, performing the following operations:

- install a component called “IdP demo” (In order to properly install the “IdP demo”, is recommended to read the following document: <https://ec.europa.eu/cefdigital/wiki/download/attachments/82772096/eIDAS-Node%20Demo%20Tools%20Installation%20and%20Configuration%20Guide%20v1.4.3.pdf>)
- set the `node.type` property, available in `nationalAdapter.properties` file, with value “eidas”. Such a value forces the National Adapter to work in a “simulate mode”.

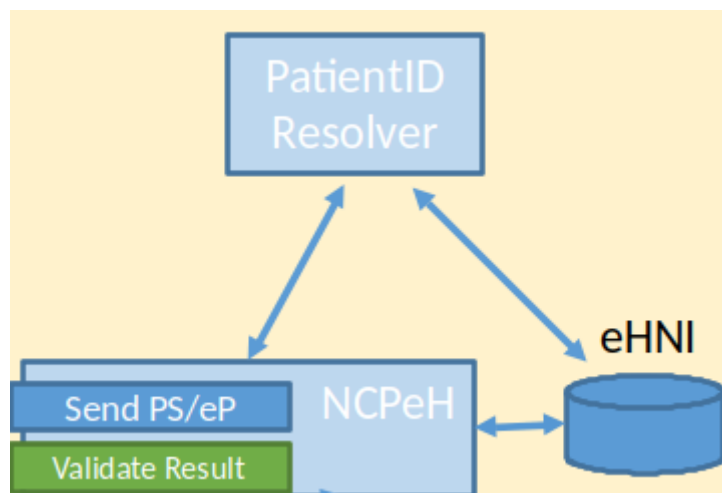
It has to be noted that, today, a Country B cannot use the above mentioned “IdP demo” in order to simulate the eIDAS flow for a Country A1 which is not yet ready with its eIDAS infrastructure and, *at the same time*, integrate a Country A2 ready for the full eIDAS experience. In other words, what can be done today is a full eIDAS experience for any Country of Origin, or a simulated eIDAS experience for any Country of Origin. We foresee this enhancement can be done at the eIDAS connector level, where the several EU flags are displayed – if, for example, the user chooses Country A2, a real eIDAS authentication flow is performed, and if Country A1 is chosen, the eIDAS stub is engaged. This improvement is out of the scope of this project and can be developed in a second stage.

All tests have been performed with Java 8 and Tomcat 8.

The following image shows the various flows and protocols engaged:



1.11 PatientID Resolver



This component is not provided by HEALTHeID, although its existence is assumed. It's up to each country how it is implemented: whether in the National Connector of NCPeH-A or in the National Infrastructure of country-A itself. It must be able unambiguously resolve whatever data it receives from the eHDSI XCPD request into a patient ID that can be returned to country-B for the subsequent cross-border eHealth requests. It should be noted that the information coming from the eIDAS world is sent as-is to the eHDSI world within the XCPD, e.g., the eIDAS PersonIdentifier will arrive in NCPeH-A in the format IT/PT/XXXXX, for an Italian citizen that authenticated against a Portuguese Service Provider.