End-to-End Encryption
Known recipient
Cookbook
Version 2.4

This document is provided to you free of charge by the

eHealth platform

Willebroekkaai 38 – 1000 Brussel

38, Quai de Willebroeck – 1000 Bruxelles

Table of contents

To the attention of: “IT expert” willing to integrate this WS.

1. Document management

1.1 Document history

Version	Date	Author	Description of changes / remarks
2.3	06/05/2011	eHealth platform	Update
2.4	18/07/2018	eHealth platform	Update

2. Introduction

2.1 Goal of the service

The End-To-End Encryption (ETEE) basic services only offer building blocks that allow integrating secure communications in applications.

It does not offer a pre-packaged 'End-To-End' business solution. This means you have to create your own client application with an implementation of:

- an ETK Client
- a KGSS Client
- a software that integrates the Crypto Library
- a way to pass on a message reference to a message receiver
- a way to pass on a key reference to a message receiver (optional if a key reference is used in the Message Storage Server (MSS))
- a Message Storage Center (you could store the message reference in the MSS).

2.2 Goal of the document

This document is not a development or programming guide for internal applications. Instead, it provides functional and technical information and allows an organization to integrate and the ETEE services in their own custom application.

This document will provide all the necessary elements to get you started developing. It explains in that context:

- the main concepts and principles
- the usage of ETK Depot WS
- the usage of the Java Crypto Library

This cookbook only deals with the first phase of the ETEE project: encryption of messages for a 'known recipient' (also known as 'addressed messages').

However, in order to interact in a smooth, homogeneous and risk controlled way with a maximum of partners, these partners must commit to comply with the requirements of specifications, data format and release processes of the eHealth platform as described in this document.

Technical and business requirements must be met in order to allow the integration and validation of the eHealth platform service in the client application.

2.3 eHealth platform document references

On the portal of the eHealth platform, you can find all the referenced documents.[1]. These versions or any following versions can be used for the eHealth platform service.

ID	Title	Version	Date	Author
1	Glossary			eHealth platform
2	Cookbook "End-to-End-Encryption for a unknown recipient"	1.4	18/07/2018	eHealth platform
3	Cookbook STS	1.2	13/04/2018	eHealth platform

[1] ***https://www.ehealth.fgov.be/ehealthplatform***

3. Support

3.1 For issues in production

eHealth platform contact center:

- Phone: 02/788 51 55
- Mail: ***support@ehealth.fgov.be***
- *Contact Form :*
  - ***https://www.ehealth.fgov.be/ehealthplatform/nl/contact*** (Dutch)
  - ***https://www.ehealth.fgov.be/ehealthplatform/fr/contact*** (French)

3.2 For issues in acceptance

Integration-support@ehealth.fgov.be

3.3 For business issues

- regarding an existing project: the project manager in charge of the application or service
- regarding a new project and other business issues: ***info@ehealth.fgov.be***

3.4 Certificates

- In order to access the secured eHealth platform environment you have to obtain an eHealth platform certificate, used to identify the initiator of the request. In case you do not have one please consult the chapter about the eHealth Certificates on the portal of the eHealth platform

  https://www.ehealth.fgov.be/ehealthplatform/nl/ehealth-certificaten

  https://www.ehealth.fgov.be/ehealthplatform/fr/certificats-ehealth
- For technical issues regarding eHealth platform certificates

  Acceptance: ***acceptance-certificates@ehealth.fgov.be***

  Production: ***support@ehealth.fgov.be***

3.5 Business Continuity Plan

To ensure the continuity of the business in case of failure of the eHealth ETKDepot service, the following should be implemented by the consumer of the ETKDepot WS:

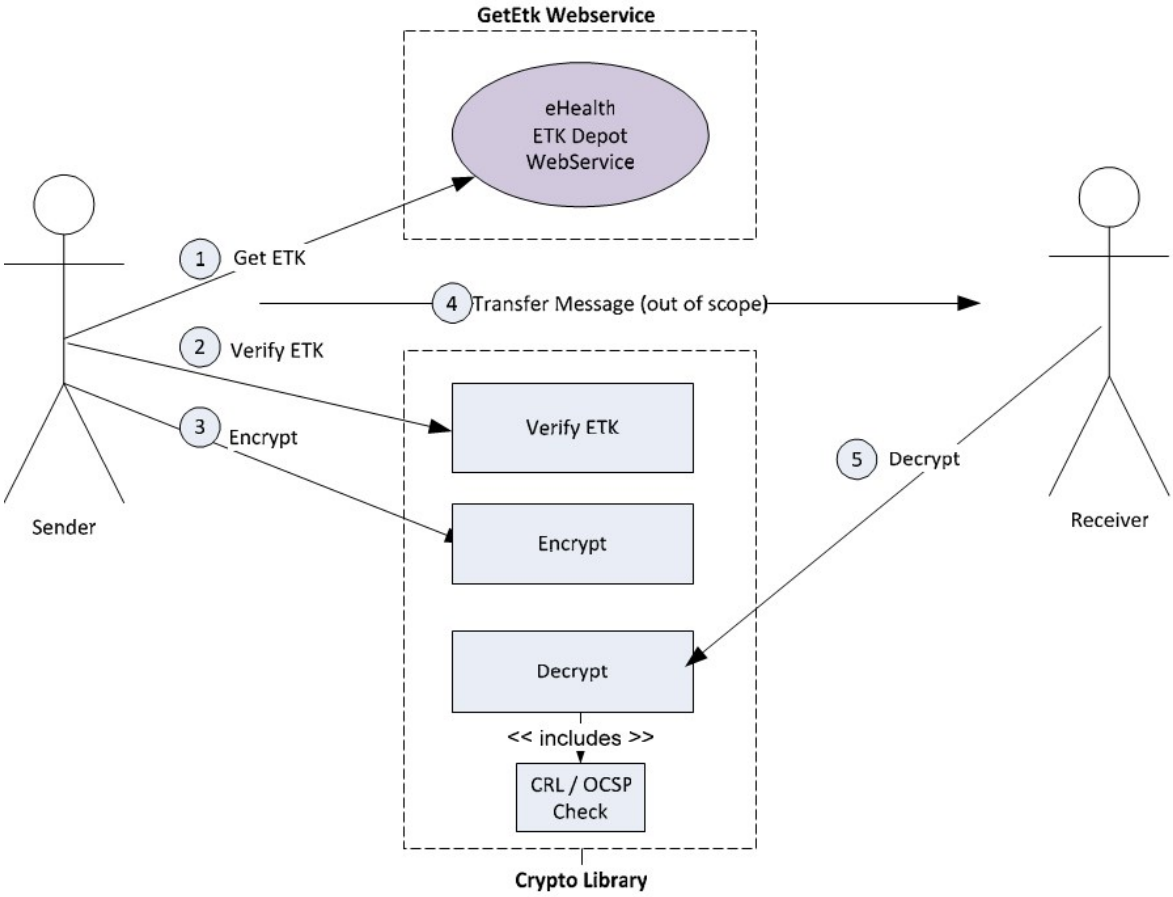
- Every consumer should cache its own ETK locally.
- The consumer should cache every response from the ETKDepot service. Only in case of unavailability of the ETKDepot service, the local cache should be used to retrieve ETKs.

4. Global overview

4.1 Schema implicating both getETK WS and Crypto Library

The ETEE service consists of two main parts:

- The first part provides a web service (WS) that will allow you to fetch the ETK of a recipient. You need the ETK of the recipient to send a message.
- The second part is the Crypto Library that will seamlessly interact with the ETK. The crypto library will seal and unseal a secure message in one easy step. It can also verify the validity of an ETK and the eHealth authentication certificate.



Sequence of events for:

The sender:

1. Searching the ETK in the ETK Depot using the getEtk WS
2. Verifying the validity of the ETK using the VerifyEtk method of the Crypto Library.
3. Securing a plaintext message using the Seal method of the Crypto Library. I.e. Signing and encrypting and signing (aka “Triple Wrapping”) and conversion of the message to the specific CMS3 format.

Transfer message:

4. Note that the ETEE base service (the ETK Depot nor the Crypto Library) does not provide tools for passing the CMS message from the sender to the receiver. A message can be passed by any possible means (email, WS, cd-rom, tape, internet, intranet, disks ...) and this process is not within the scope of the ETEE project.

The receiver:

5. Decrypting the message using the Unseal method of the Crypto Library. This includes a thorough verification (certificateChecker) of the authentication certificates being used to sign the message.

   Message authenticity and integrity is confirmed. The authentication certificate of the sender is included in each message.

All steps are performed through the Crypto Library, except the first step: the getEtk WS that fetches the ETK from the ETK Depot via WS technology.

4.2 Concepts and technologies

This section provides you with a brief introduction and some references to important concepts and technologies.

4.2.1 Asymmetric and Symmetric Encryption

The eHealth Encryption service is based on Public Key cryptography meaning that both asymmetric and symmetric encryption algorithms are used to secure information. Asymmetric encryption algorithms use, as the name somewhat suggests, different keys for encryption and decryption. Symmetric algorithms use the same key for encrypting and decrypting information.

Asymmetric encryption involves the usage of a public encryption key to encrypt information and a private encryption key to decrypt information. The private key file and its corresponding password must be protected to the maximum possible extent.

4.2.2 Triple Wrapping

Since message security relies on integrity, authenticity and confidentiality simply signing and encrypting a message does not suffice. Signing and Encrypting does not guarantee message authenticity because is vulnerable to “surreptitious forwarding”.

A triple wrapped message is a message that has been signed, then encrypted, then signed again. Further details about these principles and the reasons behind why triple wrapping is required can be found in the paper “Defective Sign & Encrypt in S/MIME,PKCS#7, MOSS, PEM, PGP, and XML“ – Donald T. Davis (MIT)

4.2.3 Certificates

A certificate binds a public key to an identity with the signature of the Certification Authority. It holds no confidential information. The private key of the corresponding public key is usually password protected and must be kept private.

End-To-End encryption uses two different key pairs for the signing and encryption to enhance security. An eHealth CA-issued authentication key pair corresponds to exactly one eHealth-issued encryption key pair. Each certificate has its public- and private key. The public keys are available in the authentication- and encryption certificate and both are available in the same ETK.

4.2.4 The ETK

An ETK holds multiple certificates (an authentication- and an encryption certificate) and has all the necessary elements to verify the integrity and identity of the holder of the private decryption key associated with the ETK.

An ETK essentially contains two certificates that are linked together: a specific certificate for encryption and your authentication certificate for signing. This encryption certificate is signed with your CA-issued authentication certificate. Additionally the ETK is signed by the eHealth RA.

When sending a secured message the ETK of the recipient must be obtained. This will give the assurance that the message is sent to someone who was properly identified by the eHealth platform. You may not cache ETKs

locally, always use ‘a fresh’ ETK. The eHealth platform will only guarantee the validity of the ETK at the time of downloading.

4.2.5 Encrypted message format

The encrypted data messages are CMS messages as specified in RRF 3852. The Crypto Library regardless of their original format can encrypt any content. A successful decryption will return the same that was originally encrypted.

5. Step-by-step

5.1 The ETK Depot

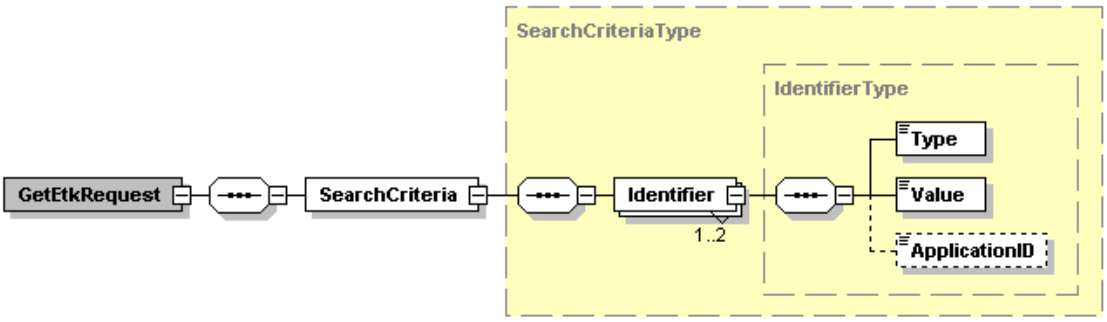
5.1.1 Accessibility

The ETK Depot is accessible via WS. The WS URL for the ETEE getEtk WS can be obtained by contacting the eHealth platform.

5.1.2 The getEtk request

This part describes the structure of a getEtk request message.

5.1.2.1 Structure of the response



The getEtkRequest contains solely an Identifier part:

Field name	Description
	The Identifier block contains the search criteria for the ETK. The search criteria are of the identifierType. Usually one identifier suffices to uniquely identify an ETK. An optional second identifier that must match also can be added. (these identifiers can be seen as conditions where the AND operator will be applied).

An identifierType is composed as follows:

Field name	Descriptions
Type	The "Type" field identifies the type of identification number based on its origin. - For RIZIV-INAMI number, use type NIHII - For RIZIV-INAMI type hospital, use type NIHII-HOSPITAL - For company number use type CBE - For pharmacy numbers recognized by the FAGG-AFMPS use type NIHII-PHARMACY - For Social Security numbers use type SSIN
Value	The "Value" field needs to contain the number of the type you specified earlier. The value is of type string and can only contain numbers. Mind to keep the leading zeroes in e.g. the CBE number.

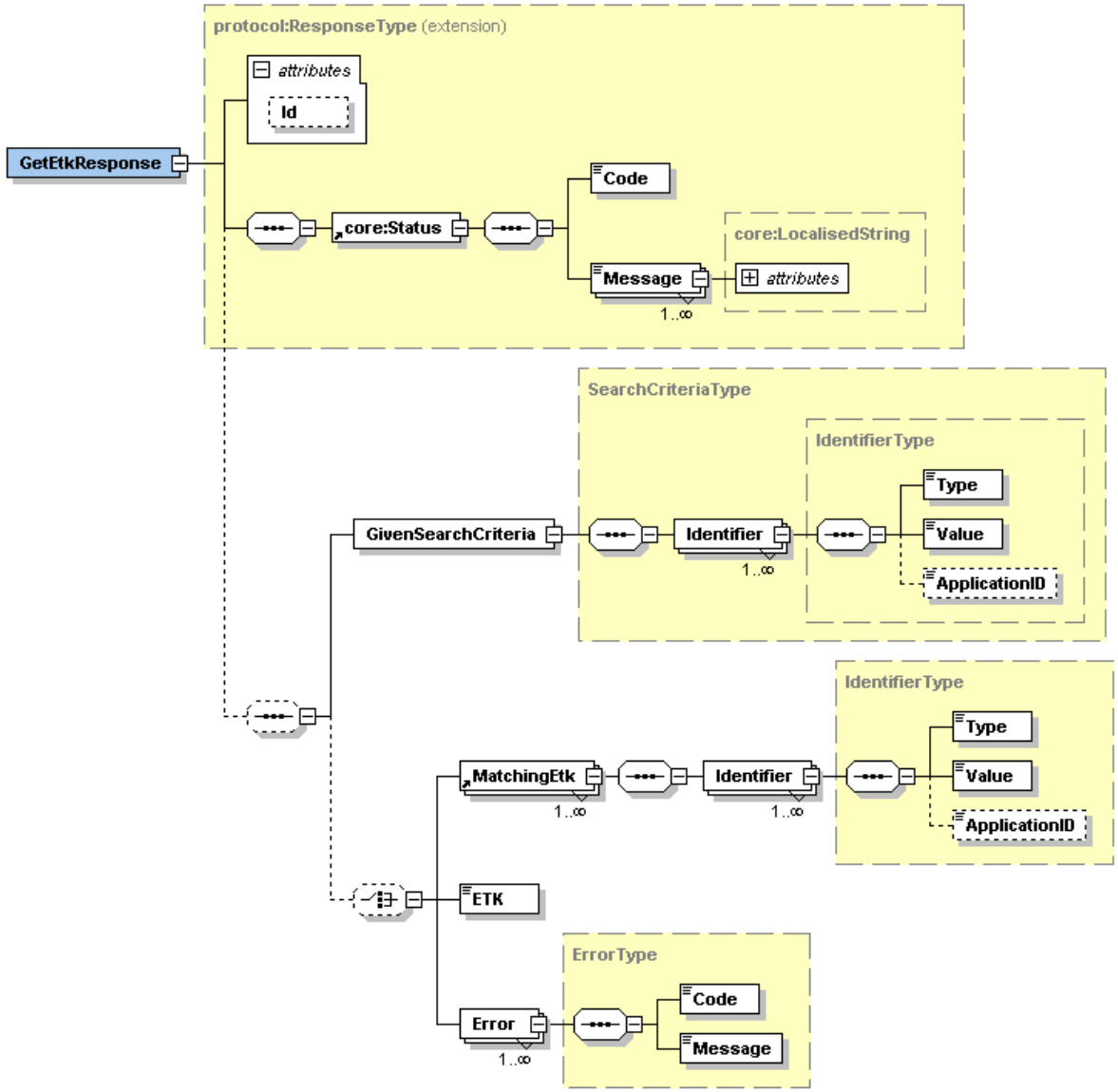
ApplicationID (Optional)	The ApplicationID field specifies a particular entity, group, section, division, software application … within the organization identified using the Type and Value fields.

5.1.3 Example of a getEtk Request

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:be:fgov:ehealth:etkdepot:1_0:protocol">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:getEtkRequ
           est>
         <urn:SearchCriteria>
            <urn:Identif
                ier>
               <urn:Type>NIHII</urn:Type>
               <urn:Value>39609058001</urn:Value>
               <urn:ApplicationID>XRays</urn:ApplicationID>
            </urn:Identifier>
         </urn:SearchCrite
                ria>
      </urn:getEtkRequ
           est>
   </soapenv:Body>
</soapenv:Envelope>
```

5.1.4 The getEtk response

5.1.4.1 Structure of the response



Field name	Description	
	The first part of the response will give internal information and a general status for your request.	
Id	The number attributed to the request/reply by the eHealth platform.	
Status	The Status block will contain a code and a message language reference.	
	Code	If no error has occurred during the transaction, the Code will be '200' - "The ETK for the given identifier has been found in the ETK Depot and is included in this response. Otherwise the status code will be:

		○ 200: The getEtkRequest was correctly processed. • 400: The getEtkRequest SOAP message is not correct. • 500: The ETK could not be retrieved due to an internal server error
	Message	Returns the message language of the message, if any.
	The second part of this request will return the search criteria you submitted in your request.	
GivenSearchCriteria	This branch if of the SearchCriteriaType in the form of one or two Identifiers used in the getEtk request. The branch is optional since it will not be displayed in case of an internal server error.	
	The following branch will return the ETK or other information when applicable. The 3 following elements are a choice.	
MatchingETK	This means that multiple ETKs have been found matching your search criteria. No ETK is actually sent when this occurs. Instead, all Identifiers of all the ETKs matching your request are returned. This will enable you to further refine your search criteria to the ETK you are required to obtain.	
ETK	Your search yielded a unique result. The ETK itself is returned in Base64 format.	
Error	A semantic error occurred. One or more error messages are included in the response.	

5.1.4.2 Example of a getEtk Response

Remark: Showing the complete contents of the <ETK> tag has no additional value. Therefore, a portion has been cut out in the middle of the string to reduce the size of this document. The original requests yields about two additional pages of output.

```
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getEtkResponse
xmlns:ns3="urn:be:fgov:ehealth:commons:1_0:core"
xmlns:ns2="urn:be:fgov:ehealth:etkdepot:1_0:protocol"
>
         <ns3:Status>
            <Code>200</Code>
            <Message Lang="EN">The getEtkRequest was correctly
processed.</Message>
         </ns3:Status>
         <ns2:GivenSearchCriteria>
            <ns2:Identifier>
               <ns2:Type>NIHII</ns2:Type>
               <ns2:Value>123</ns2:Value>
               <ns2:ApplicationID>XRays</ns2:ApplicationID>
            </ns2:Identifier>
         </ns2:GivenSearchCriteria>
<ns2:ETK>TUlBR0NTcUdTSWIzRFFFSEFxQ0FNSUFDQVFFeEN6QUpCZ1VyRGdNQ0dnVUFN
SUFHQ1NxR1NJYjNEUUVIQWFDQUpJQUVnZ1BvTUlJRUNEQ0NBdkNnQXdJQkFnSVJBTlVtc
mN0V2d5MEcrc1BWbHJicGc2a3dEUVlKS29aSWh2Y05BUUVGQlFBd2da……..
```

```
</ns2:ETK>
      </ns2:getEtkResponse>
   </S:Body>
</S:Envelope>
```

6. Risks and security

6.1 Security

6.1.1 Business security

In case the development adds an additional use case based on an existing integration, the eHealth platform must be informed at least one month in advance with a detailed estimate of the expected load. This will ensure an effective capacity management.

In case of technical issues on the WS, the partner may obtain support from the contact center (see Chap 3)

In case the eHealth platform finds a bug or vulnerability in its software, we advise the partner to update his application with the newest version of the software within 10 business days.

In case the partner finds a bug or vulnerability in the software or WS that the eHealth platform delivered, he is obliged to contact and inform us immediately. He is not allowed to publish this bug or vulnerability in any case.

6.1.2 Web service

There is no WS security on the WS and no encryption on the messages.

7. Test and release procedure

7.1 Procedure

This chapter explains the procedures for testing and releasing an application in acceptation or production.

7.1.1 Initiation

If you intend to use the eHealth platform service, please contact ***info@ehealth.fgov.be***. The project department will provide you with the necessary information and mandatory documents.

7.1.2 Development and test procedure

You have to develop a client in order to connect to our WS. Most of the required info to integrate is published on the portal of the eHealth platform.

Upon request, the eHealth platform provides you in some cases, with a mock-up service or test cases in order for you to test your client before releasing it in the acceptance environment.

7.1.3 Release procedure

When development tests are successful, you can request to access the acceptance environment of the eHealth platform. From this moment, you start the integration and acceptance tests. The eHealth platform suggests testing during minimum one month.

After successful acceptance tests, the partner sends his test results and performance results with a sample of “eHealth request” and “eHealth answer” by email to his point of contact at the eHealth platform.

Then the eHealth platform and the partner agree on a release date. The eHealth platform prepares the connection to the production environment and provides the partner with the necessary information. During the release day, the partner provides the eHealth platform with feedback on the test and performance tests.

For further information and instructions, please contact: ***integration-support@ehealth.fgov.be***.

7.1.4 Operational follow-up

Once in production, the partner using the eHealth platform service for one of his applications will always test first in the acceptance environment before releasing any adaptations of its application in production. In addition, he will inform the eHealth platform on the progress and test period.

8. Error and failure messages

Following problems and solutions are specific to setting up you configuration environment for the first time.

8.1 InvalidKeyException: Illegal key size

"InvalidKeyException: Illegal key size". "java.lang.SecurityException: Unsupported keysize or java.security. InvalidKeyException: Illegal key size"

Cause: The unrestricted policy files for the JVM you are using have not been installed.

Mitigation:

- Download and install the required files
- Follow the instructions in the README.txt
- Make sure you are installing the two new JAR files into the JVM you are running with: <java-home>/lib/security

These files can be found via ***http://java.sun.com*** at the same page where you downloaded the JDK/JRE (usually at the bottom of the page).

8.2 The Bouncy Castle security provider has not been configured

"java.security.NoSuchProviderException: no such provider: BC

at sun.security.jca.GetInstance.getService(GetInstance.java:66)

at sun.security.jca.GetInstance.getInstance(GetInstance.java:190)

at java.security.Security.getImpl(Security.java:661) … "

Cause: The Bouncy Castle security provider (implementation of abstract java.security extension) has not been installed.

Mitigation: The installation of the Bouncy Castle security provider can be done at runtime or in the java.security policy files.

To add the provider at runtime use:

import java.security.Security;

import org.Bouncy Castle.jce.provider.Bouncy CastleProvider;

Security.addProvider(new Bouncy CastleProvider());

The provider can also be configured as part of your environment via static registration by adding an entry to the java.security properties file (found in $JAVA_HOME/jre/lib/security/java.security, where $JAVA_HOME is the location of your JDK/JRE distribution). You can find detailed instructions in the file but basically it comes down to adding this line:

security.provider.<n>=org.Bouncy Castle.jce.provider.Bouncy CastleProvider

Where <n> is the preference you want the provider at (1 being the most preferred).

8.3 CertificateChecker class must be initialized

The CertificateChecker class must be initialized with a RevocationStatusChecker implementation. You can choose to:

- write and use your own implementation of the interface RevocationStatusChecker.
- use an out-of-the-box implementation such as the CrlRevocationStatusChecker: CertificateChecker.init (new CrlRevocationStatusChecker());

The CrlRevocationStatusChecker retrieves the CRL URL from the certificate that is passed as a parameter.

Currently the OcspRevocationStatusChecker is still a mock implementation, which always returns false (not revoked).

9. Licences

In order to respect the licence agreement of third party software providers, the eHealth platform is required to publish the following information:

9.1 Apache

9.2 Bouncy Castle

10. Annex

Communication regarding the JavaEnd-To-EndEncryptionLibrary and the .Net Library For End-To-End Encryption

The eHealth platform offers its users access to the Java End-To-End Encryption Library (Java ETEE Library). This library, distributed under a free license, is available on its website at the following address: ***https://www.ehealth.fgov.be/ehealthplatform***

Certain elements of the Java ETEE library of the eHealth platform use external components distributed under the Apache license and under the license distributed with the software « The Legion of The Bouncy Castle ».

In addition to the rules specified in the licenses mentioned above, the user must also take into account the following independent and additional stipulations regarding the guarantee and liability of the managers, administrators, employees and staff members of the eHealth platform.

When adapting a free software package, the eHealth platform makes every effort in order for the software to function correctly, nevertheless without assuming any obligation of result with respect to this matter.

The user commits himself to use the Java ETEE library that is available to him in the most correct and adequate way possible and to provide the eHealth platform, if necessary, with all the necessary information in order to solve problems concerning the use of the library.

Since the use of the Java ETEE library is free, the eHealth platform can on no account be held responsible for any kind of damage, direct or indirect, secondary or accessory, material or moral, caused to the user or to any third party, because of the use or the impossibility to use the library.

The Java ETEE Library must not be confused with the .Net Library for End-To-End Encryption, which was developed by Siemens on behalf of Microsoft.

The .NET ETEE Library, an adaptation of the eHealth .Net Java ETEE Library, is available on the website http://etee.codeplex.com/. The library is distributed in compliance with the terms of the GNU Lesser General Public License. It is free and available to anyone to use it. The documentation available in the library was written and published by Siemens. Users who want more guarantees can conclude a contract with Siemens or with any other service provider. In accordance with the conditions contained in this contract, the users of the .NET ETEE Library will only have access to the technical support offered by the concerned service provider.

The eHealth platform does not offer any technical support with regard to the .NET Libraries.

The eHealth platform can therefore in no event be held responsible for any damage, direct or indirect, secondary or accessory, material or moral, caused to the user or to any third party, as a result of the use or the impossibility to use the library.

Questions or remarks about the .NET ETEE Library can be posted on this website http://etee.codeplex.com/.