End-To-End Encryption Unknown recipient Cookbook Version 1.7

eHealth platform

Willebroekkaai 38 – 1000 Brussel

38, Quai de Willebroek – 1000 Bruxelles

Table of contents

Table of contents **2**

1. Document management **4**

1.1 Document history 4

2. Introduction **5**

2.1 Goal of the service 5

2.2 Goal of the document 5

2.3 eHealth document references 5

2.4 External document references 6

3. Support **7**

3.1 Helpdesk eHealth platform 7

3.1.1 Certificates 7

3.1.2 For issues in production 7

3.1.3 For issues in acceptance 7

3.1.4 For business issues 7

3.2 Status 7

4. Global overview **8**

4.1 Overview of all required components 8

4.2 High-level schema of the Unknown Recipients functionality 8

4.3 Detailing the steps 10

5. Step-by-step **12**

5.1 Technical requirements 12

5.1.1 Prerequisites 12

5.1.2 WS-I Basic Profile 1.1 12

5.1.3 Tracing 12

5.1.4 eHealth platform Authentication Certificate 12

5.1.5 ETK 13

5.2 Process overview 13

5.3 General description 13

5.3.1 ErrorType 13

5.3.2 EteeResponseType 13

5.3.3 CredentialType 14

5.4 The Key Generation Storage Services (KGSS) 15

5.4.1 GetNewKey 15

5.4.2 GetKey 18

6. Risks and security **21**

6.1 Security 21

6.1.1 Business security 21

6.1.2 Web service 21

7. Test and release procedure **22**

7.1 Procedure 22

7.1.1 Initiation 22

7.1.2 Development and test procedure 22
7.1.3 Release procedure 22
7.1.4 Operational follow-up 22
8. Error and failure messages 23
8.1 EteeResponseType 23
8.2 Error codes originating from the eHealth platform: 23

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
1.3	06/05/2011	eHealth platform	Update
1.4	17/07/2018	eHealth platform	Update
1.5	09/04/2020	eHealth platform	WS-I Compliance
1.6	22/04/2021	eHealth platform	§ 5.1.3 Tracing
1.7	19/07/2022	eHealth platform	§ 2.4 eHealth platform document references (updated) § 3.2 Status (added) § 5.1.3 Tracing (updated)

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 intended as an integration reference for the eHealth platform's ETEE basic service - Unknown Recipients. The target audience is software integrators implementing the ETEE services in their own custom application. This document is not a software manual for end users but explains the concepts, principles and interface of the KGSS WS and the Crypto Library.

The unknown recipient functionality requires the use of all the known recipient components. You need the cookbooks "ETEE known recipients" and "ETEE unknown recipients" in order to integrate the "ETEE Unknown recipients" functionality.

This document will provide you with all the necessary elements to get you started developing. In this context, it explains:

- the main concepts and principles
- the use of KGSS web services
- the use of the unknown recipients functionality offered by the Java Crypto Library.

2.3 eHealth document references

All the document references can be found on the eHealth platform portal[1]. These versions or any following versions can be used for the eHealth platform service.

ID	Title	Version	Date	Author
1	Glossary.pdf	1.0	PM	eHealth platform
3	End-to-end Encryption for a known recipient (addressed messages)"	2.9	18/07/2022	eHealth platform
4	Secure Token Service - HolderofKey	1.5	13/07/2022	eHealth platform
5	Crypto Library	1.6.1	08/2010	eHealth platform

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

2.4 External document references

All documents can be found through the internet. They are available to the public, but not supported by the eHealth platform.

ID	Title	Source	Date	Author
1	Basic Profile Version 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html	24/08/2004	Web Services Interoperability Organization

3. Support

3.1 Helpdesk eHealth platform

3.1.1 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***

3.1.2 For issues in production

eHealth platform contact centre:

- Phone: 02 788 51 55 (on working days from 7 am till 8 pm)
- 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)

3.1.3 For issues in acceptance

Integration-support@ehealth.fgov.be

3.1.4 For business issues

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

3.2 Status

The website ***https://status.ehealth.fgov.be*** is the monitoring and information tool for the ICT functioning of the eHealth services that are partners of the Belgian eHealth system.

4. Global overview

4.1 Overview of all required components

The ETEE unknown recipients service builds on the functionality of the known recipient's service:

- the ETK
- the ETK Depot
- the Crypto Library functionality for known recipients (Seal/Unseal/VerifyEtk).

Furthermore, it consists of two additional parts:

- The Key Generation Storage Service (KGSS) that creates, stores and delivers symmetrical keys. The KGSS is a service provided by the eHealth platform that does not store messages.
- The Crypto Library functionality for unknown recipients (SealForUnknown/UnsealByUnknown) that uses symmetrical encryption.

The eHealth platform does not provide a Message Storage Center (MSC), only a KGSS. The eHealth platform never stores medical information, not even encrypted.

4.2 High-level schema of the Unknown Recipients functionality

These additional WS for Unknown Recipients and Crypto Library methods need a working solution. The example below describes how to use them in a business scenario. Note that this is an example and that these steps largely depend on the implementation requirements of the eHealth platform's client.

The following high-level schema represents the steps the sender and the receiver need to carry out in order to communicate in a secure way.

The lines in bold indicate that Unknown Recipients building blocks are used. These steps will be detailed in the next section. The remaining steps are examples on how to implement the solution but they are out of scope. The eHealth platform only offers services for encrypting and decrypting information (the library) and key storage (the ETK and KGSS).

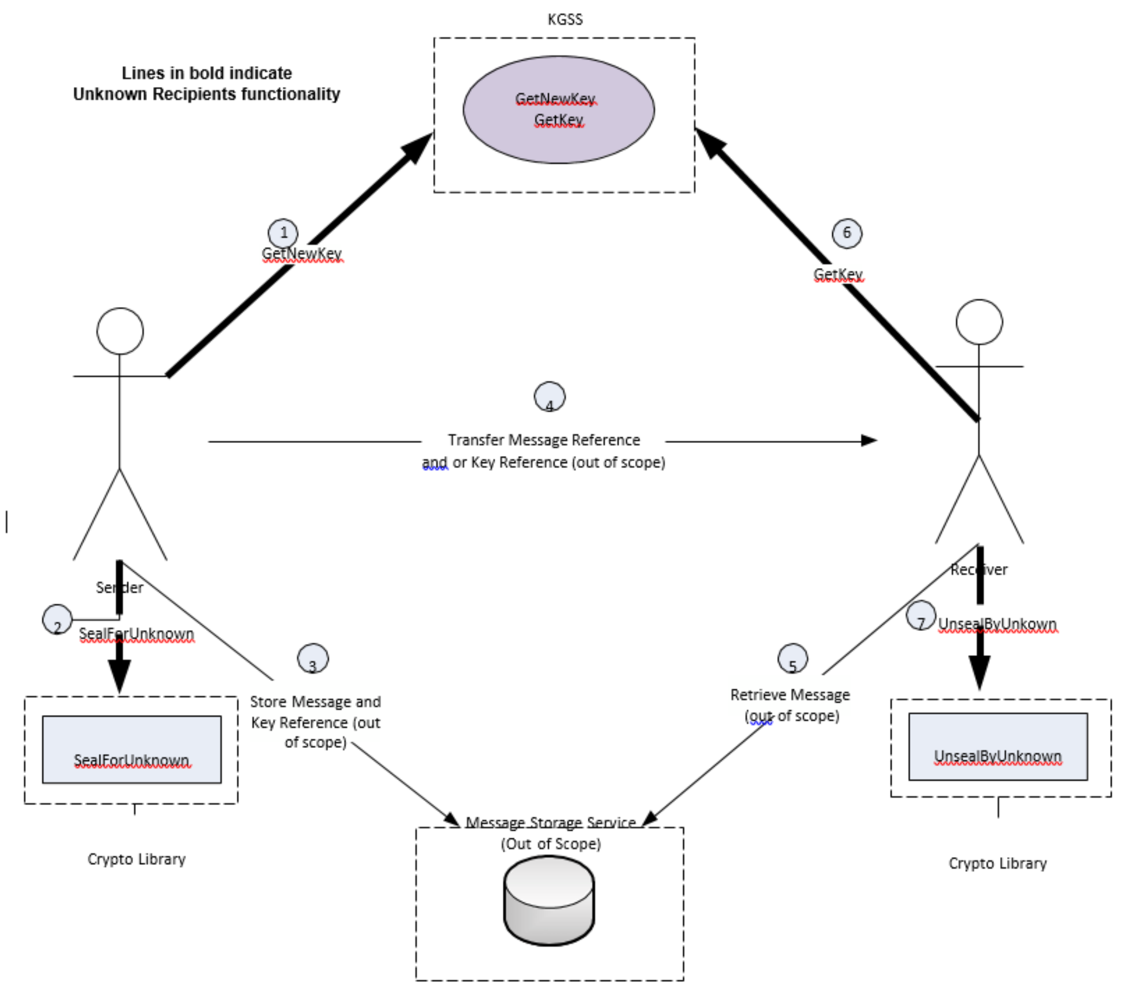


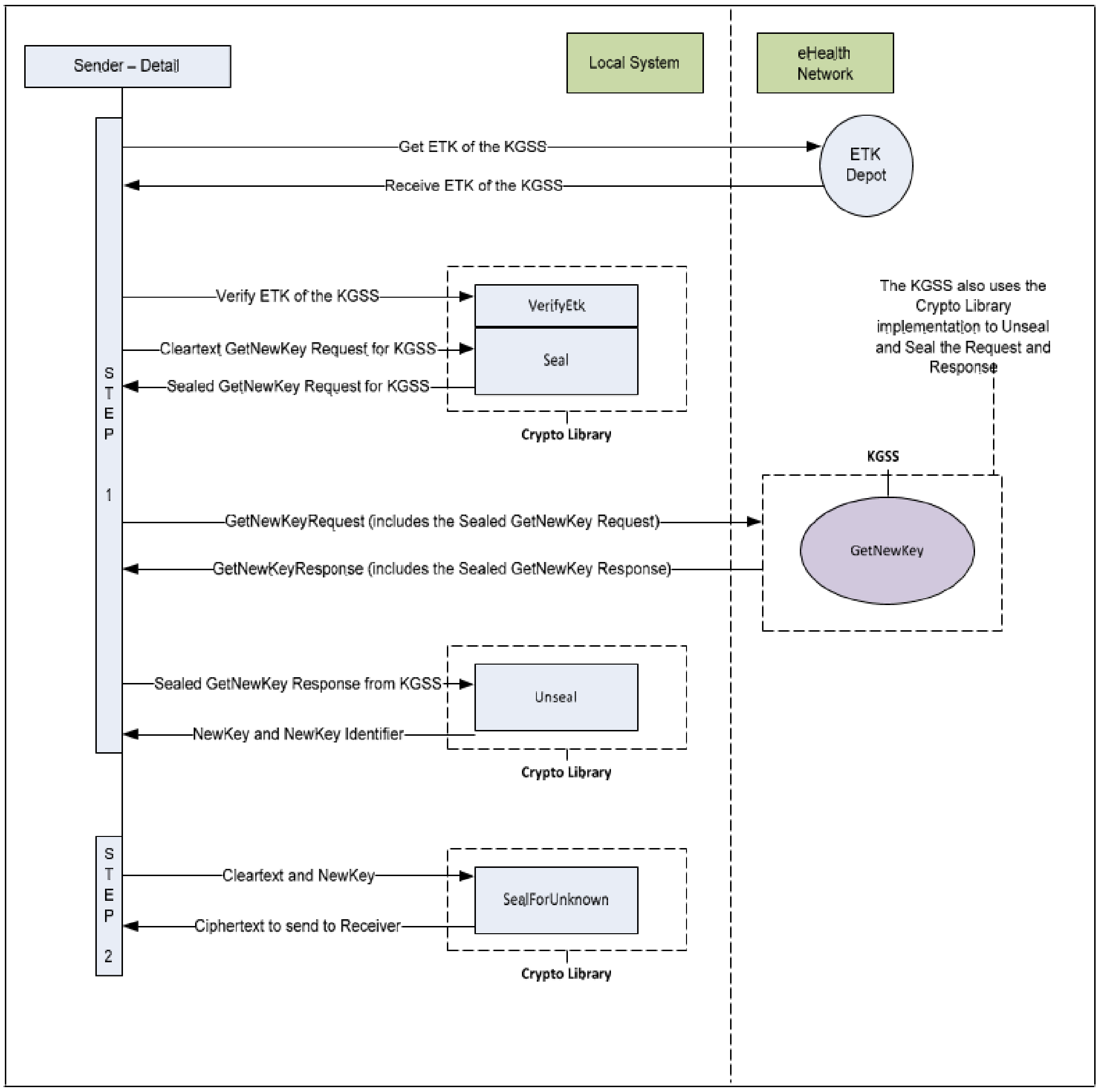
Figure 1: High-level schema

4.3 Detailing the steps

Step 1. The sender requests a new key and calls the KGSS, which returns a key and its corresponding key identifier. The secured part of the message request must be sealed with the ETK of the KGSS. The secured part of the response is sealed by the KGSS and must be unsealed by the sender.

The functionality for interaction with the KGSS is covered by the KGSS WS. (See section “5.1 Get New Key”)

This request/response communication cycle with the KGSS requires the use of the “known recipients” functionalities to encrypt/decrypt the communication between the sender and the KGSS.



Sender – Detail
Local System
eHealth Network
Get ETK of the KGSS
Receive ETK of the KGSS
ETK Depot
Verify ETK of the KGSS
Cleartext GetNewKey Request for KGSS
Sealed GetNewKey Request for KGSS
VerifyEtk
Seal
Crypto Library
The KGSS also uses the Crypto Library implementation to Unseal and Seal the Request and Response
STEP 1
KGSS
GetNewKeyRequest (includes the Sealed GetNewKey Request)
GetNewKeyResponse (includes the Sealed GetNewKey Response)
GetNewKey
Sealed GetNewKey Response from KGSS
NewKey and NewKey Identifier
Unseal
Crypto Library
STEP 2
Cleartext and NewKey
Ciphertext to send to Receiver
SealForUnknown
Crypto Library

Figure 2: Detail GetNewKey

Step 2. The sender can use the Crypto Library (implemented in his own secured system) to SealForUnknown the message by using the new key. The Crypto Library covers this functionality. (See section “6.1 SealForUnknown”

Step 3. The message must be stored in a MSS as this service is defined. It is not a building block of the ETEE unknown recipient basic service (out of scope). Therefore, the eHealth platform does NOT provide any kind of MSS nor does it provide an interface or authorization and authentication mechanisms. This system must be entirely implemented by the eHealth platform customer as a part of his project. The eHealth platform does not store these confidential messages.

Step 4. The key and/or message reference of the message is passed on to the recipient of the message. This part does not lie within the scope of the ETEE unknown recipient basic service. The key identifier and/or message reference can be passed on by any possible means (SMS, paper, email, WS, file ...).

Step 5. The receiver can retrieve the message with its message reference from the MSS.

Step 6. The receiver needs to obtain the key that corresponds to the key identifier from the KGSS, depending on which element has been provided. The functionality that deals with the interaction with the KGSS is covered by the KGSS WS. Please consult section "5.2 GetKey". This request/response communication cycle with the KGSS requires the use of the "known recipients" functionalities to encrypt/decrypt the communication between the receiver and the KGSS. If the receiver has both the key reference and the message reference available to him, he can invert steps 5 and 6.

Step 7. The retrieved message can be decrypted by using the 'UnsealByUnknown' method of the Crypto Library, which requires the symmetrical key and the encrypted message as input arguments. The Crypto Library covers this functionality. Please consult section "6.2 UnsealByUnknown.

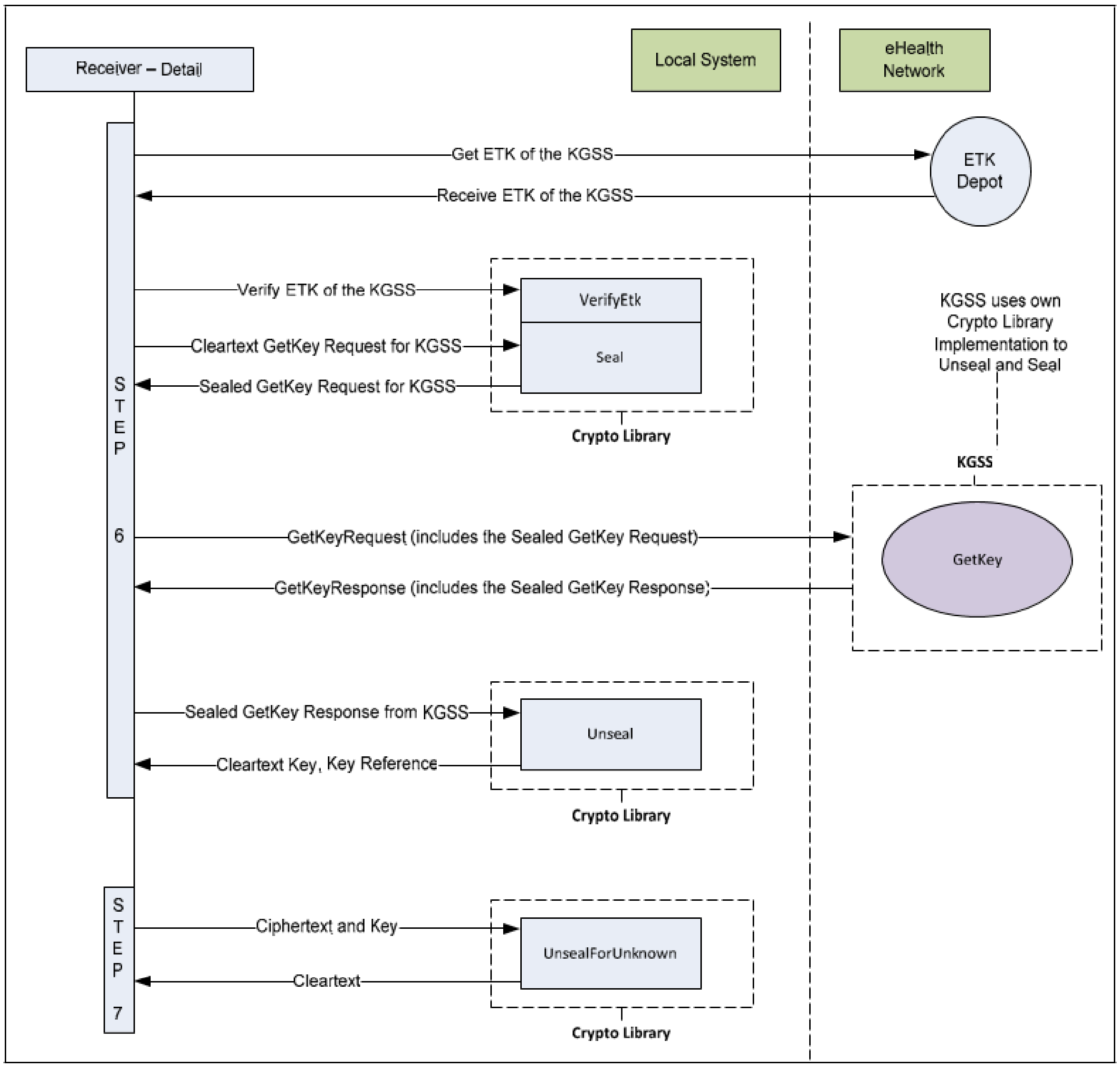

Receiver – Detail
Local System
eHealth Network
Get ETK of the KGSS
Receive ETK of the KGSS
ETK Depot
Verify ETK of the KGSS
Cleartext GetKey Request for KGSS
Sealed GetKey Request for KGSS
VerifyEtk
Seal
Crypto Library
KGSS uses own Crypto Library Implementation to Unseal and Seal
KGSS
STEP 6
GetKeyRequest (includes the Sealed GetKey Request)
GetKeyResponse (includes the Sealed GetKey Response)
GetKey
Sealed GetKey Response from KGSS
Cleartext Key, Key Reference
Unseal
Crypto Library
STEP 7
Ciphertext and Key
Cleartext
UnsealForUnknown
Crypto Library

Figure 3: Detail GetKey

5. Step-by-step

5.1 Technical requirements

5.1.1 Prerequisites

5.1.1.1 Java Cryptography Extension (JCE)

You must download and install the Java(TM) Cryptography Extension Unlimited Strength Jurisdiction Policy Files 5.0.

https://cds.sun.com

5.1.1.2 Bouncy Castle

The library has been tested with Bouncy Castle 1.39 or higher for Java version 1.5.0. The required Bouncy Castle libraries are delivered with the ETEE package. You can download the required JARs can from the Legion of the Bouncy Castle.

www.bouncycastle.org

5.1.1.3 Get Log 4j

This component is required to show exactly what the Crypto Library is doing when used, so that this log information can be shown or saved.

Apache log4j is a Java based logging utility. You must use version 1.2 or higher. The software can be downloaded from the Apache Software Foundation.

www.junit.org
logging.apache.org

5.1.2 WS-I Basic Profile 1.1

Your request must be WS-I compliant (Cfr External Ref). If not you will receive one of the errors SOA-03001 – SOA-03003.

5.1.3 Tracing

To use this service, the request SHOULD contain the following two http header values (see RFC ***https://datatracker.ietf.org/doc/html/rfc7231#section-5.5.3***):

1. User-Agent: information identifying the software product and underlying technical stack/platform. It MUST include the minimal identification information of the software such that the emergency contact (see below) can uniquely identify the component.
    a. Pattern: {minimal software information}/{version} {minimal connector information}/{connector-package-version}
    b. Regular expression for each subset (separated by a space) of the pattern: [[a-zA-Z0-9-\/]*\/[0-9azA-Z-_.]*
    c. Examples:
       User-Agent: myProduct/62.310.4 Technical/3.19.0
       User-Agent: Topaz-XXXX/123.23.X freeconnector/XXXXX.XXX
2. From: email-address that can be used for emergency contact in case of an operational problem.
   Examples:
   From: ***info@mycompany.be***

5.1.4 eHealth platform Authentication Certificate

To use the ETEE service, you must have an authentication certificate from the eHealth platform. Please contact our helpdesk if you have any doubts about the compatibility of your current certificate.

5.1.5 ETK

In order to secure communications between a client and the KGSS, an ETK is required. Please see the instructions in the "Requesting eHealth Certificates" document.

5.2 Process overview

In order to use the building blocks that are specific to unknown recipients, the following actions are required:

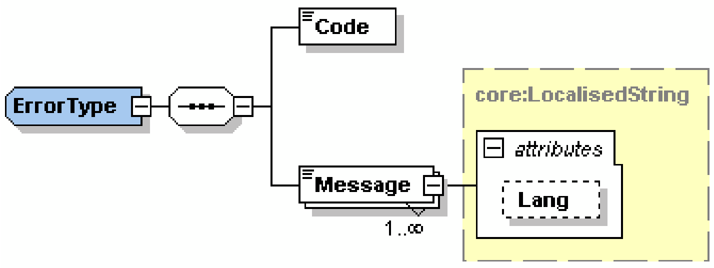
1. Integrate and comply with the prerequisites (Java, JCE, bouncy castle).
2. Obtain an eHealth Authentication Certificate. The procedure 'Requesting eHealth Certificates' can be found on the eHealth platform portal.
3. Obtain an ETK (using the ETEE Requestor application). The procedure 'Requesting eHealth Certificates' can be found on the eHealth platform portal. In order to use the 'Unknown Recipients' functionality, you must also have an ETK for secure communications from and to the KGSS.
4. The Unknown Recipients functionality is based on the Known Recipients functionality. Therefore, you need the Known Recipients components and functionality:
   - ETK Depot (getEtk);
   - Crypto Library (Seal/Unseal/VerifyEtk).
5. Integrate the unknown recipient's functionality:
   - KGSS
   - Crypto Library version 1.5 or above (SealForUnknown/UnsealByUnknown)

5.3 General description

The following section explains some complex elements defined as a type and used as re-usable definitions in the KGSS WS.

5.3.1 ErrorType

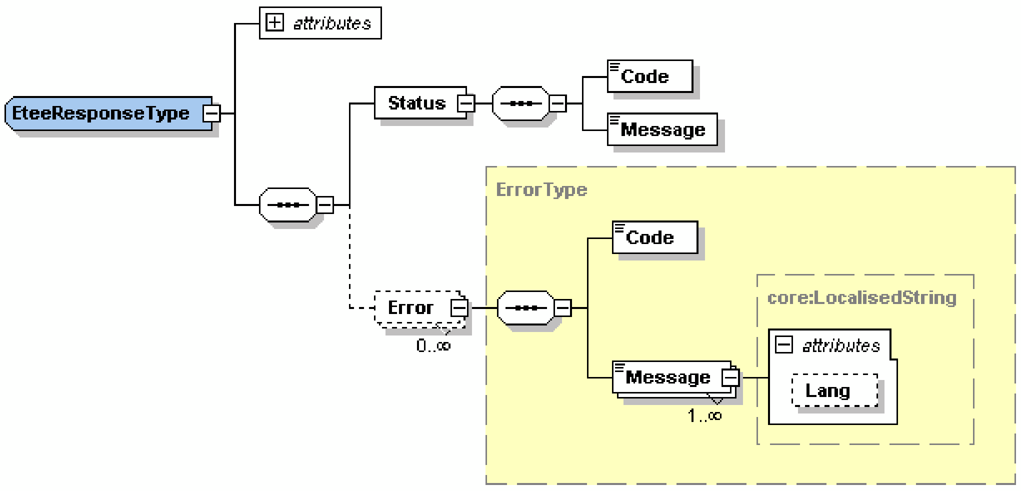
If an error occurs, the following information will be displayed.



Field name	Description
Code	A custom code has to be defined. Three-letter error code.
Message	The error message. The 'Lang' attribute contains the language of the error

5.3.2 EteeResponseType

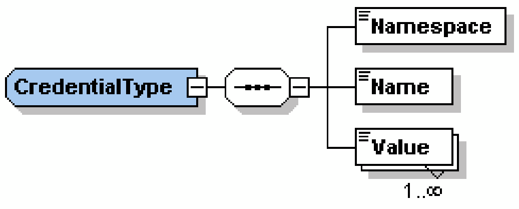
The EteeResponseType is a custom type used to provide generic information with every response.
This standard way of providing feedback information enhances readability.



Field name	Descriptions	
Status	The Status block will contain a code and a message language reference.	
	Code	
	Message	A three-letter error code that defines the status of the request.
Error (optional)[2]	This custom type is also discussed in detail in the 'Types' section of this document. See section "4 General ".	

5.3.3 CredentialType

A credentialType is used within the ETEE project to identify an AllowedReader or an ExcludedReader. These credentials are structured as follows:



Field name	Description
Namespace	The namespace of the Credential attribute, e.g.- "urn:be:fgov:ehealth:certified-namespace"
Name	The name of the Credential attribute, e.g. - "urn:be:fgov:ehealth:doctor-nihii"
Value	The value of the Credential attribute, e.g.- "74042015445"

[2] *This field is optional because a response does not always contain an error. Setting this field as mandatory would result in an invalid response for each correctly processed request.*

5.4 The Key Generation Storage Services (KGSS)

The KGSS is only accessible through WS. The URL can be obtained by contacting the eHealth platform. You must have an authentication certificate from the eHealth platform in order to access the service. There are two basic functionalities, both described in detail and explained in relation to their position in the high-level overview schema.

All KGSS communication is encrypted:

1. requests from the message sender or receiver to the KGSS are sealed by using the ETK of the KGSS
2. responses from the KGSS to the message sender or receiver are sealed by using the sender's or receiver's ETK.

Remark:

The ETK of the KGSS should be retrieved using the ETKDepot[3] WS.

The following search criteria must be given to request the ETK of the KGSS:

- Type: CBE
- Value: 0809394427
- ApplicationId: KGSS

All of these steps must be completed for each message:

- GetNewKey – see step 1 on Figure 1: High-level schema - the sender calls the KGSS with a request to return him a new key and an identifier, used by the sender to encrypt information by using the SealForUnknown method of the Crypto Library.

- E.g. the sender requests a new key. The KGSS creates and stores a new key. The sender receives a response with "AFAA18926FDF65C367BF9A838DAB4EF3" as the key identifier and"EE37154F94DBF8F8D42E218397B3EA24" as the key.
- GetKey – see step 6 on Figure 1: High-level schema - the receiver of the sender's actual encrypted message can call the KGSS, asking for a specific existing key once he has obtained the key reference3. Therefor he can use the key's 'key identifier' as a reference. The KGSS returns the corresponding key, to be used by the receiver of the message in order to decrypt the information by using the Crypto Library's UnsealByUnknown method.
  E.g., the sender requests the key for the previously created key identifier "AFAA18926FDF65C367BF9A838DAB4EF3".
  The KGSS returns the key "EE37154F94DBF8F8D42E218397B3EA24" in the response.
- Some examples of Key Identifiers and Keys that are stored in the KGSS: Each 'Key Identifier' is unique and corresponds to exactly one 'Key'.

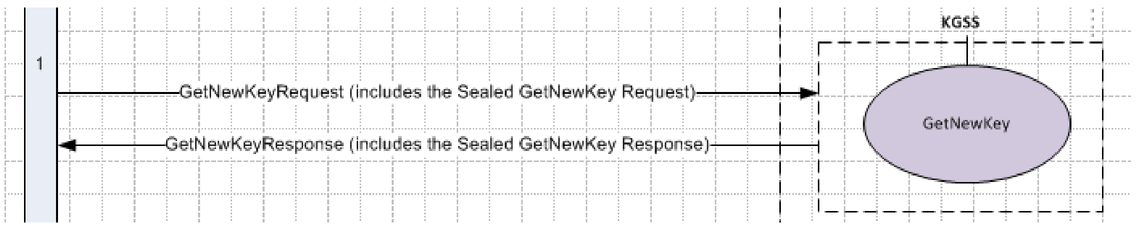
Key Identifier	Key
AFAA18926FDF65C367BF9A838DAB4EF3	EE37154F94DBF8F8D42E218397B3EA24
201E606A1EDA1B61414B6A746A1417D0	852C061A622857776F3CB3313318E2A2
883CDBA88E2B981A531556719838A769	6F838BF8D8F84C70B87B529E10AE4C2F

Please note that depending on the size of the secured information, the size of the 'SealedContent' elements shown in the different examples will be larger.

5.4.1 GetNewKey

Step 1 of the high level overview.

This method will create and store a new symmetrical encryption key, used by the sender. The key and its identifier are returned to the requestor.

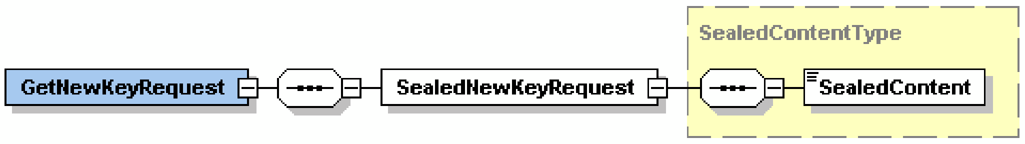

1
GetNewKeyRequest (includes the Sealed GetNewKey Request)
GetNewKeyResponse (includes the Sealed GetNewKey Response)
KGSS
GetNewKey

Detail from functional step 1 in the high-level overview: get a new key from the KGSS by the sender.

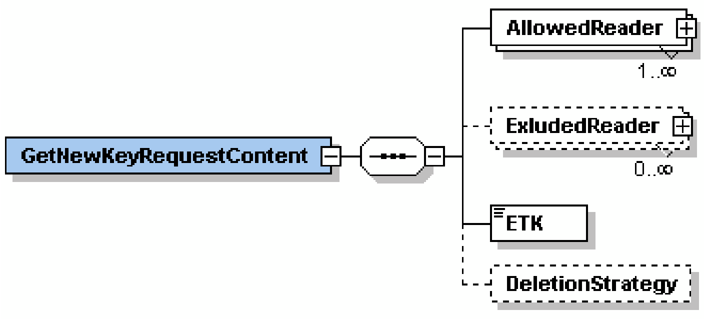
5.4.1.1 Requesting the new key

Every Unknown Recipient message sent goes with a request for a new key.

5.4.1.1.1 Structure of the GetNewKeyRequest



The GetNewKeyRequest contains the following information:

Field name	Descriptions	
	------- The Following information is Encrypted for the KGSS by the sender -------------- -  GetNewKeyRequestContent AllowedReader 1..∞ ExludedReader 0..∞ ETK DeletionStrategy	
SealedNewKeyRequest / SealedContent	AllowedReader	A list of AllowedReaders. This list contains the readers, allowed to obtain the newly created key. The allowed readers must be identified with the eHealth CredentialType. This custom type is discussed in detail in the 'Types' section of this document. See section "4 General information" E.g. All of type doctor and one dentist with NIHII='1234567890'
	ExcludedReader (optional)	A list of ExcludedReaders. This list contains the readers that are explicitly excluded from getting the newly created key. The excluded readers must be identified with the eHealth CredentialType. This custom type is discussed in detail in the 'Types' section of this document. E.g. For all doctors except for an excluded family member that is a doctor with NIHII='5678910123'

	ETK	The ETK provided by the requestor is used by the KGSS to encrypt the response. This is usually the ETK of the message sender.
	DeletionStrategy(optional)	Reserved for later use to define in which circumstances a key can be deleted. No further details available at the time.
	END Encrypted information	

5.4.1.1.2 Example

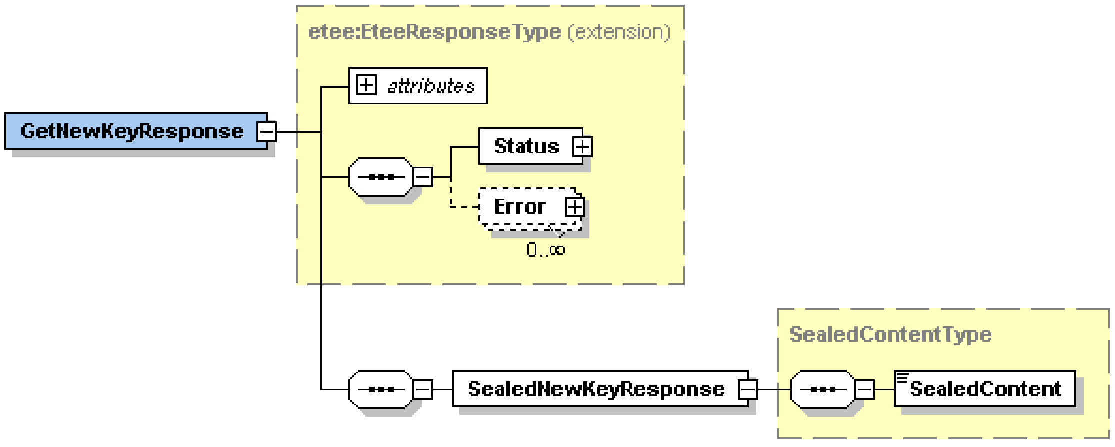
```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 sp2 (http://www.altova.com)-->
<GetNewKeyRequest xsi:schemaLocation="urn:be:fgov:ehealth:etee:kgss:1_0:protocol
ehealth-etee-kgss-schema-protocol-3_1.xsd"
xmlns="urn:be:fgov:ehealth:etee:kgss:1_0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SealedNewKeyRequest><SealedContent>UjBsR09EbGhjZ0dTQUxUjBsR09EbNQ
UFBUUNBRU1tQ1p0dU1GUXhEUzhi</SealedContent>
        </SealedNewKeyRequest
                    >
</GetNewKeyRequest>
```

5.4.1.2 *Receiving the newly generated key*

This is the response to the message sender by the KGSS.

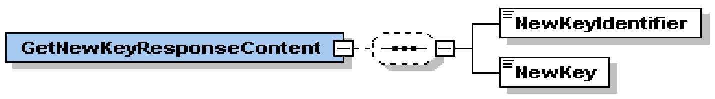
5.4.1.2.1 Structure of the response

This is a response to a GetNewKeyRequest. The client of the WS will receive the new symmetrical key and its identifier. The information is secured with the ETK that was sent in the request.



The GetNewKeyResponse contains the following information:

Field name	Description
EteeResponseType (extension)	This custom type is described in detail in the 'Types' section of this document. See section "4 General information".
SealedNewKeyResponse / SealedContent	The Following information is Encrypted with the ETK given in the Request

		
	NewKeyIdentifier	Contains the identifier (=reference) for the new key to be passed onto the recipient.
	END Encrypted information	

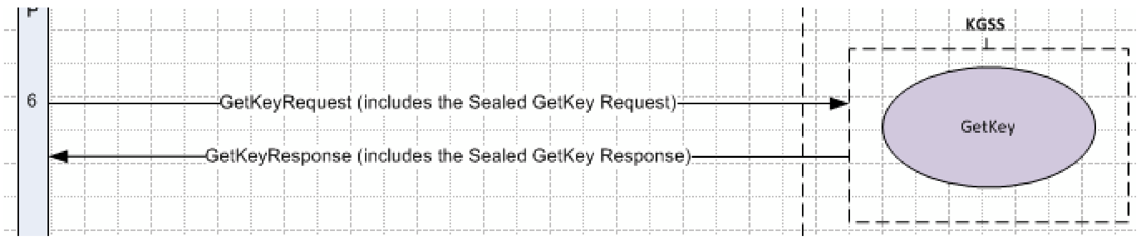
5.4.1.2.2 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<n1:GetNewKeyResponse Id="String" xsi:schemaLocation="urn:be:fgov:ehealth:etee:kgss:1_0:protocol
ehealth-etee-kgss-schema- protocol-3_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:n1="urn:be:fgov:ehealth:etee:kgss:1_0:protocol">
        <Status><Code>200</Code><Message> The KGSSRequest was correctly
processed.</Message></Status>
        <n1:SealedNewKeyResponse>
        <n1:SealedContent>RU1tQ1p0dU1GUXhEUzhiUjBsR09EbGhjZ0dTQUxNQUFBU
UNB</n1:SealedContent>
        </n1:SealedNewKeyResponse>
</n1:GetNewKeyResponse>
```

5.4.2 GetKey

This method will retrieve an existing symmetrical encryption key, used by the receiver. The key identifier must be provided in the request and the symmetrical key itself is returned in the response. For this request, you must have a valid SAML token (See Cookbook STS).

It is a part of step 6, previously described in the example scenario: the part in which the communication with the KGSS is turned into a request/response cycle.

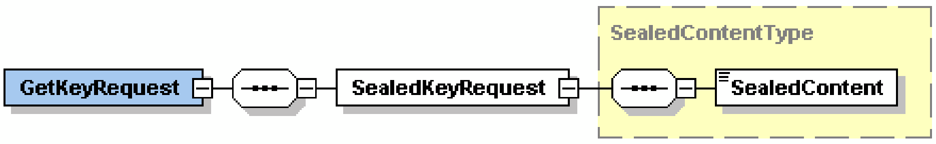

KGSS
6
GetKeyRequest (includes the Sealed GetKey Request)
GetKeyResponse (includes the Sealed GetKey Response)
GetKey

(Detail from functional step 6 in the high-level overview: get an existing key from the KGSS by the receiver).

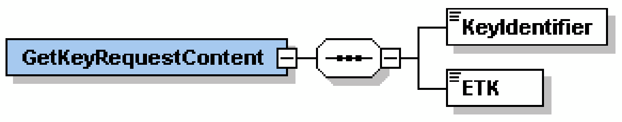
5.4.2.1 Retrieving an existing key

For each Unknown Recipient message, a key must be fetched from the KGSS.

5.4.2.1.1 Structure of the request



The GetKeyRequest contains the following information:

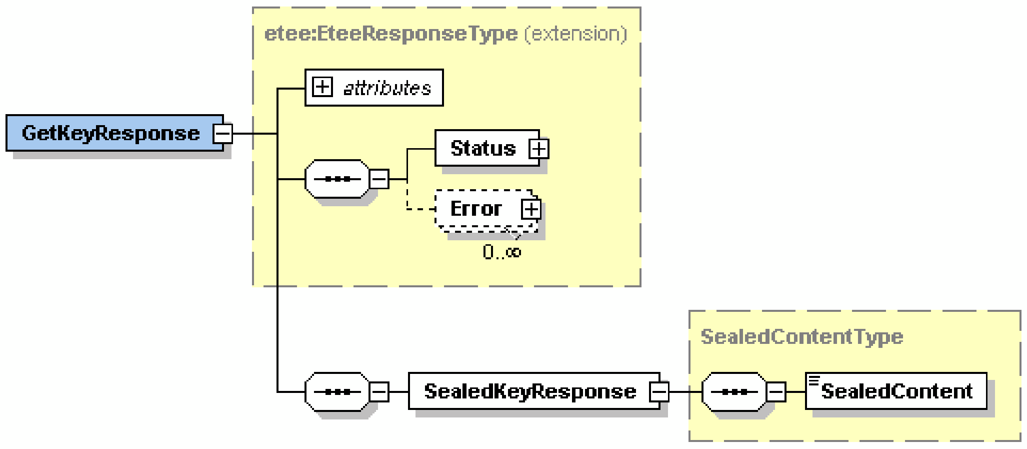
Field name	Description
	The following information is Encrypted for the KGSS by the sender.
SealedKeyReques/Sealed Content	
	KeyIdentifier: The identifier of the key that must be retrieved
	ETK: The ETK of the requestor used by the KGSS to encrypt the response. This is usually the ETK of the message receiver
	End Encrypted information

5.4.2.1.2 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<GetKeyRequest xsi:schemaLocation="urn:be:fgov:ehealth:etee:kgss:1_0:protocol  ehealth-
etee-kgss-schema-protocol-3_1.xsd" xmlns="urn:be:fgov:ehealth:etee:kgss:1_0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <SealedKeyRequest><SealedContent>0dU1GUXhEUzhiUjBsR09EbGhjZ UjBsR
0dTQUxNQUFBUUNBRU1tQ1p </SealedContent>
        </SealedKeyRequest>
</GetKeyRequest>
```

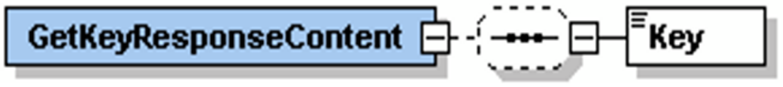
5.4.2.2 The GetKeyResponse

5.4.2.2.1 Structure of the response



The GetKeyResponse contains the following information:

Field name	Descriptions
EteeResponseType (extension)	This custom type is discussed in detail in the 'Types' section of this document. See section "4 General information".

SealedKeyResponse / SealedContent	The Following information is Encrypted with the ETK given in the Request 	
	Key	In order to obtain the required symmetrical key, you must decrypt your message by using the Crypto Library UnsealByUnknown method.
	-------------------------------- END Encrypted information ------------------------------------	

5.4.2.2.2 Example of GetKeyResponse message

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2007 sp2 (http://www.altova.com)-->
<n1:GetKeyResponse Id="String" xsi:schemaLocation="urn:be:fgov:ehealth:etee:kgss:1_0:protocol
ehealth-etee-kgss-schema- protocol-3_1.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:n1="urn:be:fgov:ehealth:etee:kgss:1_0:protocol">
<Status><Code>200</Code><Message> The KGSSRequest was correctly
processed.</Message></Status>
            <n1:SealedKeyResponse>
         <n1:SealedContent>GhjZ0dTQUxUjBsR09EbNQUFBUUNBRU1tQ1p0dU1GUXhE
Uzhi</n1:SealedContent>
            </n1:SealedKeyResponse>
</n1:GetKeyResponse>
```

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.

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

6.1.2 Web service

WS security used in this manner is in accordance with the common standards. Your call will provide:

- SSL one way
- Time-to-live of the message: one minute.
- Signature of the timestamp, body and binary security token. This will allow eHealth to verify the integrity of the message and the identity of the message author.
- No encryption on the message.

7. Test and release procedure

7.1 Procedure

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

Once a KGSS client has been developed that can connect to our WS in the acceptance environment, integration and acceptance tests can begin. We ask to perform tests for at least one month. The reason behind this is to protect the production environment to the fullest against any incidents.

When developing an application that will use the ETEE infrastructure, you must notify the eHealth platform by a written document at least 3 months before the scheduled production date. This is required for efficient capacity planning. It will allow the eHealth platform to assure the eHealth SLAs. Please refer to the contact center for more information.

If everything is correct, the eHealth platform and the partner agree on a release date. The eHealth platform should prepare the connection to the production environment and provide the production environment URL.

During the release day in acceptance, the partner in the health care sector provides feedback on the release test results to the designated eHealth platform contact(s). You will be provided with a list of designated contact(s) for your project.

A contract must be drawn by which you confirm as a partner that the integrator will also observe the eHealth platform rules in case of new releases of his software.

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 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 the point of contact at the eHealth platform.

The following items must be checked before you can access the production environment:

- request New keys from the KGSS (GetNewKey)
- get existing Keys from the KGSS (GetKey).

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

8.1 EteeResponseType

The EteeResponseType has status codes. These status codes can be:

- 200 The KGSSRequest correctly processed.
- 400 The KGSSRequest SOAP message incorrect.
- 500 The KGSSRequest could not be completed due to an internal server error.

8.2 Error codes originating from the eHealth platform:

These error codes first indicate a problem in the arguments sent, or a technical error.

Error code	Component	Description	Solution
SOA-03001	*Consumer*	*This is the default error for content related errors in case no more details are known.*	***Malformed message***
SOA-03002	*Consumer*	*Message does not respect the SOAP standard.*	***Message must be SOAP***
SOA-03003	*Consumer*	*Message respects the SOAP standard, but body is missing.*	***Message must contain SOAP body***