END-TO-END ENCRYPTION
Unknown recipient (KGSS) - REST
Cookbook
Version 1.2

This document is provided to you free of charge by the

eHealth platform

Willebroekkaai 38
38, Quai de Willebroek
1000 BRUSSELS

Table of contents

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

1. Document management

1.1 Document history

Version	Date	Author	Description of changes / remarks
1.0	28/11/2019	eHealth platform	Initial version
1.1	09/04/2020	eHealth platform	WS-I Compliance
1.2	22/04/2021	eHealth platform	Tracing

2. Introduction

2.1 Goal of the service

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

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

- KGSS Client
- software that integrates a cryptographic solution
- way to pass on a message reference to a message receiver
- way to pass on a key reference to a message receiver (optional if a key reference is used in the Message Storage Server (MSS))
- 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 support guide for the eHealth platform's REST service "ETEE - Unknown Recipients". Software integrators implementing the ETEE REST service in their own custom application are the target audience.

This document is not a software manual for end users but explains the concepts, principles and interface of the KGSS REST WS.

Unlike the KGSS SOAP service, the KGSS REST service does NOT require the use of the known recipient components.

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
2	eHealth Services – Web Access	2.0	12/07/2019	eHealth platform
3	I.AM Connect Technical specifications	1.1	12/08/2019	eHealth platform
4	I.AM Connect – Client Registration	1.02	25/02/2019	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 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 I.AM Connect

- In order to use the KGSS REST service you have to obtain an “Access token” which is delivered through I.AM Connect. You can find more information about I.AM Connect and how to register a client in I.AM Connect on the IAM eHealth portal page:

  Dutch version: ***https://www.ehealth.fgov.be/ehealthplatform/nl/service-iam-identity-access-management***

  French version: ***https://www.ehealth.fgov.be/ehealthplatform/fr/service-iam-identity-access-management***

4. Global overview

4.1 High-level schema of the ETEE Unknown Recipients functionality

The example below describes how to use the ETEE KGSS Rest service in a business scenario.

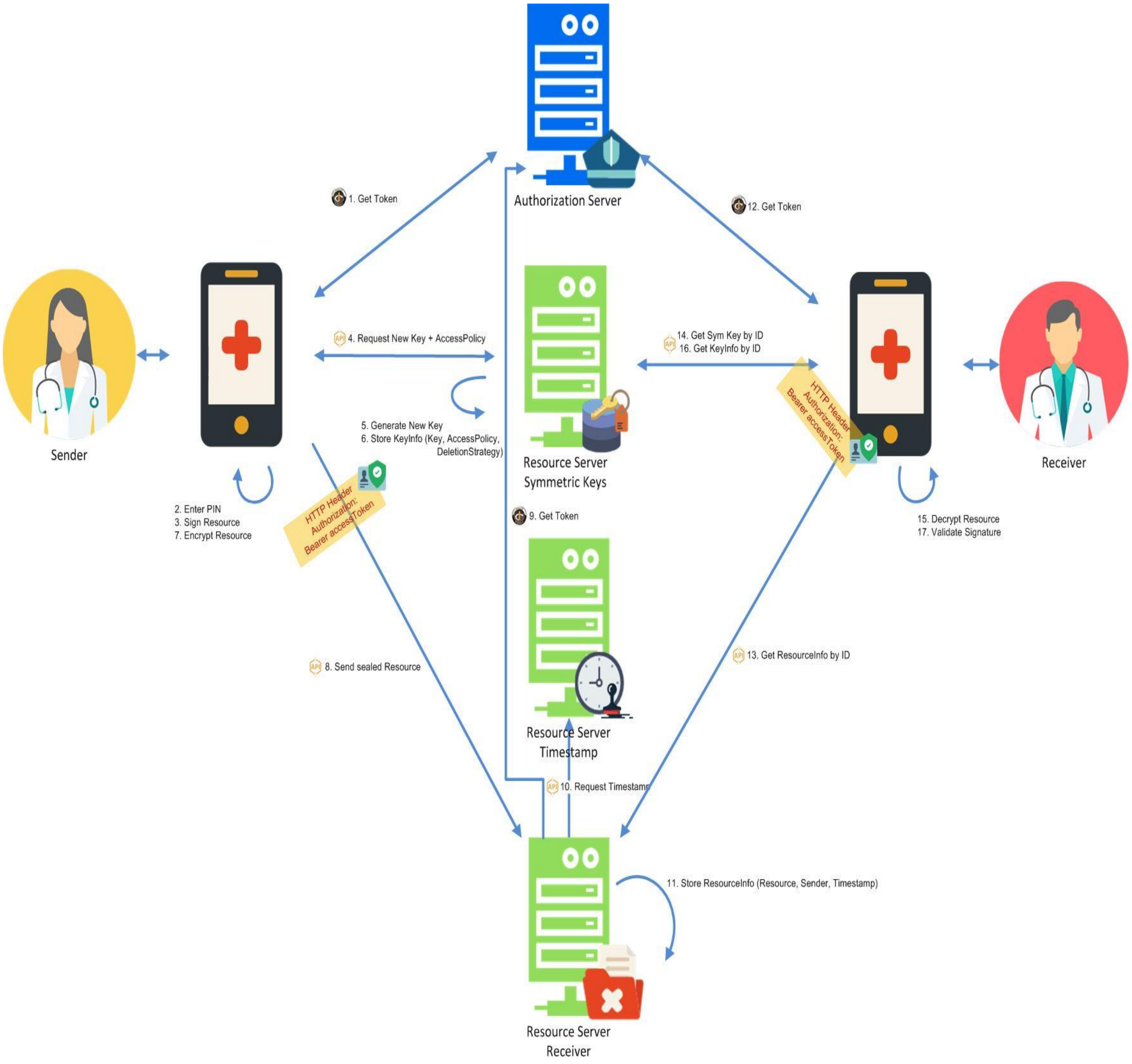

Figure 1: High-level schema

1. The client requests authorization at the eHealth platform.
   See authentication/authorization flows described in document “eHealth Services – Web Access[2]”.
2. The client requests the user to unlock his private key.
3. The client signs the message for content integrity and message authentication.
4. The client requests a symmetric key and defines who should get access to it.
5. The eHealth platform generates a new key.
6. The eHealth platform stores the new key, its identifier, the access policy and a deletion strategy and returns the key with its identifier.
7. The client encrypts the message with the key(s) and adds the reference to the key in clear.
8. The client sends the sealed message to the depot. The access token, received at the end of step 1, is added to the request.
9. If the access token contains sufficient privileges, the depot will accept the message and add a timestamp on it for proof of reception. For this, the depot requests a token on behalf of itself (client credentials grant, see authentication/authorization flows for a full description) to get permission to request a timestamp.
10. The depot requests a timestamp on the received message.
11. The depot will store the message, the identity of the sender of the message and the timestamp as proof of reception.

    How the receiver is notified that a message is available for him at the depot is out of scope of this document and can be decided per project that uses a depot between sender and receiver.
12. The client of the receiver requests authorization at the eHealth platform on his behalf. See authentication/authorization flows for a full description.
13. The client gets the message by its ID from the depot. The access token, received in previous step, is added for authentication/authorization.

    The depot will also send the identity of the sender and the timestamp, linked to the message. The format may be project-dependent and is outside of the scope of this document.
14. The client gets the symmetric key from the eHealth platform. The access token is added so the access policy can be applied based on that.
15. The client decrypts the resource.
16. To validate the signature on the message and to authenticate the owner of the key used for it, the receiver can get the key from the eHealth platform using the reference mentioned in the signature. The time of the timestamp returned by the depot can be added to be able to retrieve a key that is no longer active now but was at the time the message was sent. To verify that the sender was also the author of the message, the key must also be referenced in the identity of the sender. This verifies that the entity that sent the message to the depot is also the author, that the key was active at the time the message was sent and that the receiver was the intended audience of the original author.

    The timestamp, placed on the message in step 8 and stored in step 9, gives the receiver sufficient proof that the message was received in that state at that exact time. If such proof is not available, the receiver should use the current time when validating the key and signature on the message.
17. The receiver validates the signature.

[2] *https://www.ehealth.fgov.be/ehealthplatform/fr/search?q=Web+Access*

5. Step-by-step

5.1 Technical requirements

5.1.1 eHealth platform Authentication

As explained previously, to use the ETEE Rest service, you must have an access token delivered through I.AM Connect.

Two roles are defined for the using of the KGSS Rest service:

- read-keys: This role must be present in the access token in order to use the GET method of the service
- manage-keys: This role must be present in the access token in order to use the POST and DELETE methods of the service

<u>Presentation of the roles in the access token:</u>

```
"ehealth-etee-backend": {
  "roles": [
    "read-keys",
    "manage-keys"
  ]
}
```

For the organizations using the KGSS Rest service, the local manager of the organization has the possibility to assign these roles to his members through the UMAN application.

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://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.8***):

1. **User-Agent**: information identifying the software product and underlying technical stack/platform.

   - Pattern: {company}/{package-name}/{version} {platform-company}/{platform-package-name}/{platform-package-version}
   - Regular expression for each subset (separated by a space) of the pattern: [[a-zA-Z0-9-\/]*\/[0-9a-zA-Z-_.]*
   - Examples:
     User-Agent: MyCompany/myProduct/62.310.4 eHealth/Technical/3.19.0
     User-Agent: Topaz-XXXX/123.23.X Taktik/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

5.2 The Key Generation Storage Rest Services (KGSS)

The REST interface is described with a JSon / Swagger API.

5.2.1 POST /symkey

This method allows creating a symmetric key, defining who should get access to it, and defining a deletion strategy for this key.

5.2.1.1 Request

Element	Description
purpose	The context of the key creation. The values are fixed by the eHealth platform on basis of the business cases. Currently, two values are available: - recipe - chapter IV
allowedReaders	The allowed readers will be used to know exactly who can retrieve the key by using the secured GET operation. They are defined with claims. See “Annex 1 – Claim structure” for more information about the allowed readers definition. In order to avoid inconsistent allowed readers, there is a specific mapping between the purposes and the claims of the allowed readers. See “Annex 2 – Mapping claims/purposes” for more information about that.
deletionStrategy	The goal of the deletion strategy is to limit the access to a key during a specific duration. If an expirationDate is not specified for a key, a default duration of 1 year will be used

Example:

```
{
  "purpose":"recipe",
  "allowedReaders":[
    {
      "organizations":[
        {
          "pharmacy":{
            "recognised":"true"
          }
        }
      ]
    },
    {
      "organizations":[
        {
          "hospital":{
            "recognised":"true"
          }
        }
      ]
    },
    {
      "persons":[
        {
          "physician":{
            "nihii11":"18334780004"
          }
        }
      ]
    },
    {
      "persons":[
        {
          "ssin":"89051016482"
        }
      ]
    }
  ]
}
```

5.2.1.2 Response

Element	Description
keys	The new key generated by the backend. The kid returned is the identifier used in the GET and DELETE methods. Note: If you want to use this kid in the GetKey method of the KGSS SOAPUI service, you have to transform it from hexadecimal to base64 (e.g. *C6FF1B99FE0DE65FBBB2BBE7415FED8D -> xv8bmf4N5l+7srvnQV/tjQ==)*

Http code 201 returned in case of success

Example:

```
{"keys": [{
  "kty": "oct",
  "use": "enc",
  "kid": "C6FF1B99FE0DE65FBBB2BBE7415FED8D",
  "k": "0S6sR8K5lHwlNsILLlVjXw",
  "alg": "A256KW"
}]}
```

5.2.2 GET /symkey

This method allows to retrieve an existing key on the condition that the user has an access token containing a profile which matches the allowed readers of the key (defined during the key creation).

5.2.2.1 Request

Element	Description
keyIdentifier	The identifier of the key that you want to retrieve. This Key Identifier (kid) is returned in the response of the POST operation

5.2.2.2 Response

Element	Description
keys	The key information linked to the kid inserted in the request

Http code 200 returned in case of success

Example:

```
{"keys": [{
  "kty": "oct",
  "use": "enc",
  "kid": "C6FF1B99FE0DE65FBBB2BBE7415FED8D",
  "k": "0S6sR8K5lHwlNsILLlVjXw",
  "alg": "A256KW"
}]}
```

5.2.3 DELETE/symkey

This method allows deleting an existing key (logical delete in database). A user can only delete a key that he has created, so he cannot delete a key created by another user.

5.2.3.1 Request

Element	Description
keyIdentifier	The identifier of the key that you want to delete. This kid is returned in the response of the POST operation

5.2.3.2 Response

Http code 204 returned in case of success

6. Risks and security

6.1 Risks & safety

6.2 Security

6.2.1 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.

6.2.2 The use of username, password and token

The username, password and token are strictly personal and are not allowed to transfer.
Every user takes care of his username, password and token and is forced to confidentiality of it. Every user is also responsible for every use, which includes the use by a third party, until the inactivation.

7. Implementation aspects

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 integration 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

8.1 Http codes

Here are the error status codes that can be returned by the KGSS Rest service:

- 400 The KeyDepot REST message is incorrect.
- 401 The requestor cannot consult or delete the key
- 404 No key found linked to kid of the KGSS request
- 500 The KGSS request 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***

9. Annex 1 – Claim structure

This section explains the structure of the claims that allow to define the allowed readers.

To consult the specific list of the existing claims, see “Annex 2– Mapping claims/purposes”

9.1 Citizen

To define a citizen as allowed reader, the claim must only contain the SSIN of the individual.

```
{
      "persons": [{
        "ssin": "{$ssin}",
 }]
```

9.2 Professional

To define a professional as allowed reader, you have to specify several information:

- The quality of the professional (required)
- The NIHII11 of the professional (required)
- The claim “recognised” is optional and allow to specify that the professional must also have a valid VISA

```
{
      "persons": [{

        "{$quality}": {
           "nihii11": "{$nihii11}",
           "recognised": "true"
        }
      }]
 }
```

Here is the list of professional qualities currently available:

- physician
- dentist
- nurse
- practical_nurse

9.3 Organization

There are two possibilities to define organizations as allowed readers:

1) Define a specific organization
In this case you have to specify the following information:
    - The quality of the organization (required)
    - The id type of the organization (required)
    - The identifier of the organization (required)

```
{
        "organizations": [{
                "{$quality}":{
                        "{$idType}": "{$identifier}"
                }
            }]
},
```

2) Define all organizations of a specific quality

   In this case, you have to specify the quality that must have the organizations to be considered as an allowed reader

```
{

        "organizations": [{
            "{$quality}": {
                "recognised": "true"
            }
        }]
    },
```

This is the list of organization qualities currently available:

- hospital (id type: nihii)
- pharmacy (id type: nihii)
- enterprise (id type: cbe)
- hio (id type: cbe)
- ehp (id type: ehp)

10. Annex 2 – Mapping claims/purposes

Claim Name	Purpose	Corresponding eHealth saml attribute (KGSS Soap Service)
$."persons".["physician"].nihii11	recipe	urn:be:fgov:person:ssin:ehealth:1.0:nihii:doctor:nihii11
$."persons".["dentist"].nihii11	recipe	urn:be:fgov:person:ssin:ehealth:1.0:nihii:dentist:nihii11
$."persons".["practical_nurse"].nihii11	recipe	urn:be:fgov:person:ssin:ehealth:1.0:nihii:practicalnurse:nihii11
$."persons".["nurse"].nihii11	recipe	urn:be:fgov:person:ssin:ehealth:1.0:nihii:nurse:nihii11
$."organizations".["hospital"].nihii	recipe	urn:be:fgov:ehealth:1.0:hospital:nihii-number
$."persons".ssin	recipe	urn:be:fgov:person:ssin
$."organizations".["hospital"].recognised	recipe	urn:be:fgov:ehealth:1.0:hospital:nihii-number:recognisedhospital:boolean
$."organizations".["pharmacy"].recognised	recipe	urn:be:fgov:ehealth:1.0:pharmacy:nihii-number:recognisedpharmacy:boolean
$."persons".["physician"].nihii11	recipe	urn:be:fgov:person:ssin:ehealth:1.0:doctor:nihii11
$."persons".["physician"].recognised.[“true”]	recipe	
$."organizations".["enterprise"].cbe	chapterIV	urn:be:fgov:kbo-bce:organization:cbe-number
$."organizations".["hio"].recognised	chapterIV	urn:be:fgov:kbo-bce:organization:cbe-number:ehealth:1.0:hio:boolean