eHealthBox REST
Cookbook
Version 1.2

This document is provided to you free, of charge, by the

eHealth platform

Willebroekkaai 38 – 1000 Brussel

38, Quai de Willebroek – 1000 Bruxelles

Table of contents

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	29/11/2019	SMALS eHealthBox team	Initial version
1.1	05/12/2020	eHealth platform	Review + corrected version
1.2	21/04/2020	eHealth platform	WS-I Compliance + Business error code 810
1.3	13/11/2020	SMALS eHealthBox team + eHealth platform	New Swagger interface version + adding details description
1.4	08/02/2021	SMALS eHealthBox team + eHealth platform	Added chapter 8.2 + correction of business error 801 + added intro to chapter 8.3 + added explanation in chapter 2.1 + Complete Review + verification and adaptation of annexes
1.5	22/04/2021	eHealth platform	4.4 Technical requirements
1.6	12/07/2021	SMALS eHealthBox team	New mandatory field in annexesMetadata: contentId (also added in 9.1 java snippet)
1.7	11/10/2021	eHealth platform	Number of annexes reduced to 25 maximu**m**
1.8	01/12/2021	eHealth platform	- p.23 : viewDateTime explanation - p. 24 : binsent messages delete criterion corrected - p.24 : sent messages criterion corrected - p.34 : readDateTime explanation
1.9	22/02/2022	SMALS eHealthBox team	Lifetime of messages changed: chapter 1.17.3.3 Remarks
1.10	22/06/2022	SMALS eHealthBox team	Expiration dates added for each folder
1.11	21/09/2022	SMALS eHealthBox team	Update of error message 400 § 2.3 eHealth document references (Id 7 - updated) § 4.4.2 Tracing (updated)
1.12	03/04/2023	SMALS eHealthBox team	Curl example of request message with attachment
1.2	24/03/2026	eHealth platform	Add tracing section and update contact center hours

2. Introduction

2.1 Goal of the service

The eHealthBox REST service (**previously called eHealthBox V4**) allows an authenticated user to consult and publish (possibly encrypted) eHealthBox messages for different addressees, with optional elements such as annexes and metadata.

The eHealthBox central systems receive and process the publication request **asynchronously**. This means that a successful response does not guarantee the message will be correctly published at the end of the process. A publication failure can occur later due to the behaviour of external systems. In this case, a structured notification message *(an eHealthBox message)* is returned to the sender. A successful response message only guarantees that the message will be processed.

A user can get general information on his eHealthBox, a list of messages for a specific folder and the content of a specific message. He can also move a message to his inbox and handle his Out-of-Offices (OoO). Nevertheless, if the user has access to more than one box, it will only be possible to use one box at a time.

The size of a message is currently limited to **30MB**. The global size of an eHealthBox is limited to **10MB** by default *(including inbox and bin folder*). The maximum number of annexes is currently limited to **25**. If a box is full, the following messages will be in standby queue. Note that an encrypted message weighs more due to the encryption overhead.

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 the eHealthBox REST service in its 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 eHealthBox
- technical information about calling the service

This information should allow (the IT department of) an organization to integrate and use the WS call. 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 all the requirements the eHealth platform has described in this document.

In addition, our partners in the healthcare sector must also comply with the business rules of validation and integration of data within their own applications in order to minimize errors and incidents. In other words, 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 ones, can be used for the eHealth platform service.

ID	Title	Version	Date	Author
1	SOA – Error guide	1.0	10/06/2021	eHealth platform

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

2	Request test case template	3.0	22/02/2018	eHealth platform
3	WSDL	N.A.	N.A.	eHealth platform
4	eHealth Services – Web Access	2.0	12/07/2018	eHealth platform
5	eHealthBox Supported Qualities	1.3	28/05/2020	eHealth platform
6	eHealth platform SSO eHealthBox	1.2	22/04/2020	eHealth platform
7	End-to-Ed Encryption Known recipient Cookbook	2.9	18/07/2022	eHealth platform

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

2.5 Service history

This chapter contains the list of changes made to the service with respect to the previous version.

Previous version	Previous release date	Changes
1.0 CURRENT VERSION	24/01/2021	First major REST version : same functionalities as the eHealthBox SOAP 3.1 service but in REST version

2.6 New in this REST version

- Services are merged : there is only one service to publish and consult eHealthBox messages.
- The web service is now in a **REST** API style *(Representational State Transfer)*, easily usable through the JSON file format.
- No more 'News' in this version (only one type of messages).
- No more publications to an entire category meaning that it is no longer possible to send a message/publication to all the members of the same quality category.

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 8 am till 6 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***

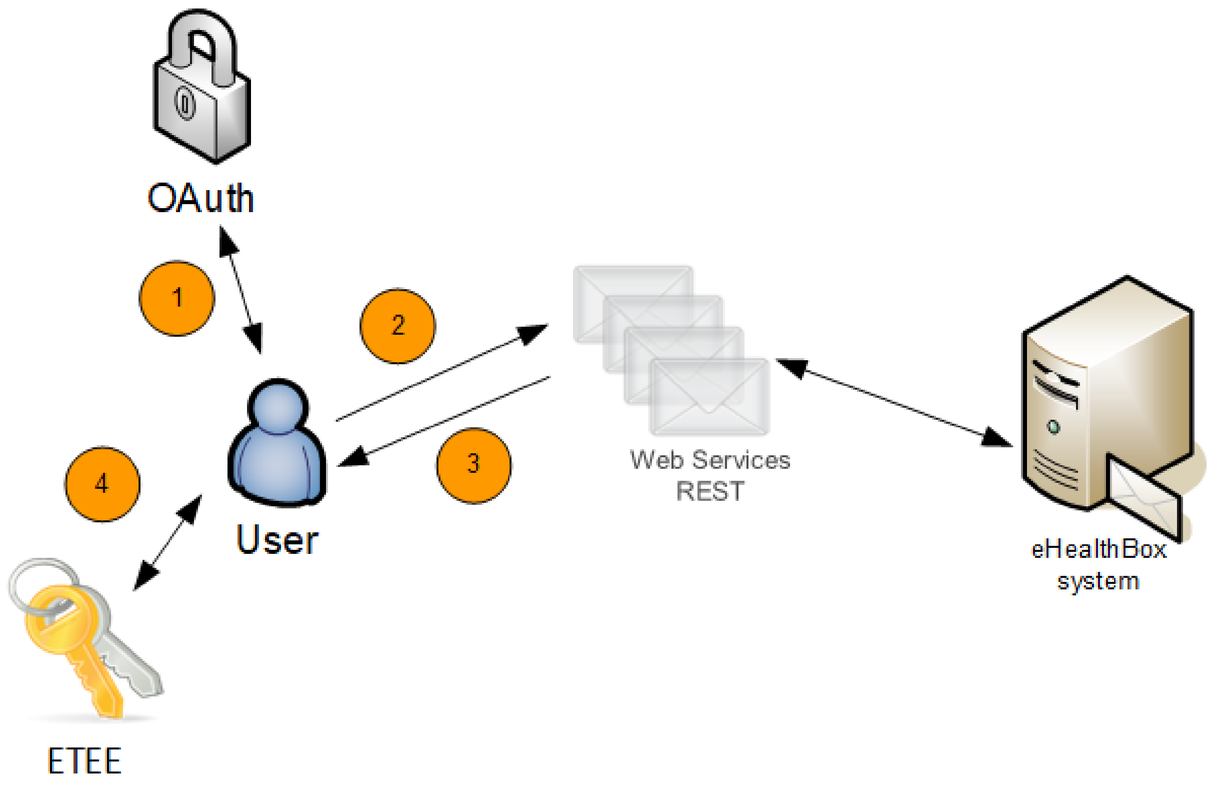
3.2 Status

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 Flow overview on how to contact the eHealthBox

This global overview shows how the endpoints are being used.



1. To use the endpoints, you first have to contact our oAuth service to get a secure token containing the identification of the user. (See Chap 2.4 -Ref 4 – eHealth Services – Web Access)
2. Secondly, thanks to his token, the user can use endpoints to manage his eHealthBox, get information on it, publish or retrieve his message(s).
3. Depending on the request of the user, the endpoints will provide the user with an answer concerning his eHealthBox or one of his eHealthBoxes or even publish messages.
4. Finally, the endpoints support encryption of the content of your message, the client encrypts/decrypts his message with his private key and the Crypto Library. (see Chap 2.4 – Ref 7 – End-to-End Encryption Known recipient Cookbook).

4.2 REST and JSON

SOAP and REST are two API styles that approach data transmission from a different point of view. SOAP is a standardized protocol sending messages using other protocols such as HTTP and SMTP.

As opposed to SOAP, REST is not a protocol but an architectural style. The REST architecture lays down a set of guidelines you need to follow if you want to provide a RESTful web service, for example, stateless and the use of HTTP status codes. As SOAP is an official protocol, it comes with higher complexity, it requires more bandwidth and resources that can lead to slower page load times.

REST was created to address the problems of SOAP. Therefore, it has a more flexible architecture. It allows different messaging formats, such as HTML, JSON, XML, and plain text, while SOAP only allows XML.

REST is also a more lightweight architecture, so **RESTful web services have a better performance**.

The REST architecture allows API providers to deliver data in multiple formats such as plain text, HTML, XML, YAML, and JSON, which is one of its most loved features. Thanks to the increasing popularity of REST, the lightweight and human-readable JSON format has also quickly gained traction, as it is an easy-to-parse and lightweight data-interchange format. In spite of its name, JSON is completely language-agnostic, so it can be used with any programming language, not just JavaScript. JSON files consist of collections of name/value pairs and ordered lists of values that are universal data structures used by most programming languages. Therefore, JSON can be easily integrated with any language.

4.3 Endpoints

The last version of REST interface described with a JSON / Swagger API is available on the eHealth API Portal :

ACC: ***https://portal-acpt.api.ehealth.fgov.be/***

PROD: ***https://portal.api.ehealth.fgov.be/***

4.4 Technical requirements

4.4.1 WS-I Basic Profile 1.1

Your request must be WS-I compliant (See Chap 2.4 - External Document Ref 1).

4.4.2 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***

Exception for Browser-Based Clients

Browsers control the User-Agent header natively and may block client-side modification for security reasons. When operating in a browser-based environment, supplying the custom User-Agent header is **not required**. That said, it **must** be included whenever the environment permits it.

The From header carries no such restriction and **must** be present in all requests, regardless of environment.

5. API description

5.1 Mailbox authentication

5.1.1 POST /mailboxes

Create mailbox when not existing or get Accesskey

The ***createMailbox*** method is used to create a new mailbox in case the requester is authenticated by the authentic source and the mailbox does not exist. If the mailbox already exists, only the accessKey will be returned.

Technical description:

The requester can create the mailbox that is linked to his JSON web token (JWT). As request body, he can either provide a mailbox identifier with the following fields (see table below), or an empty body. If no request body is provided, the profile from the token will be used and the mailbox for that profile will be created.

5.1.1.1 Request

Element	Description
entity	eHealthBox's identification number. This is a digital number representing an INSS, NIHII, EHP or CBE.
entityType	the type of identification number of the care provider
quality	eHealthBox's Quality *(See Annex 3 - eHealthBox Supported Qualities)*

Example:

```
POST https://api-acpt.ehealth.fgov.be/ehBox/mailboxes
{
    "entity": "12345678910",
    "entityType": "INSS",
    "quality": "DENTIST"
}
```

5.1.1.2 Response

Element	Description
key	The key contains the AccessKey generated based on the request attributes.
mailboxIdentifier	The mailbox Identifier contains the entity, entityType and quality from request.

Example:

```
{
   "key": "6891f2e75965315a07970e270ee04ea8",
   "mailboxIdentifier": {
      "boxIdentifiers": {
         "entity": "12345678910",
         "entityType": "INSS",
         "quality": "DENTIST"
```

```
        }
    }
}
```

5.1.1.3 *Remarks*

- An access key is unique and deterministic *(based on the entity, entityType and quality)*.
- The mailbox quota is taken from the eHealthBox default for the specified quality (configured by the eHealth administrator).
- Request inputs must be valid (checked).

5.2 Consultation

5.2.1 GET /mailboxes/{accessKey}

Consult information from a mailbox

The *getBoxInfo* method allows an authenticated user to receive general information about his mailbox.

The request can be made for any authorized mailbox by specifying the proper accessKey matching the mailbox the user can access.

5.2.1.1 *Request*

Path parameters

Path parameter	Description
accessKey	The accessKey is a unique key identifier which is generated in the functionality create mailbox or get accessKey.

Example:

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/59e2eb3e964ba857578a5c0a41efe142
```

5.2.1.2 *Response*

The response contains a success status code and general information on the eHealthBox as explained below. Attention should be paid to standbyMessagesCount larger than zero and if currentSize > quota.

Element	Description
creationTms	Date of creation of the box
lastAccessTms	Date of last access
accessKey	Object containing key and mailboxIdentifier
accessKey/Key	The accessKey from the path parameter
accessKey/mailboxIdentifier	Object with the object boxIdentifiers
mailboxIdentifier/boxIdentifiers	The identification number and the type of the eHealthBox The quality of the mailbox owner
boxIdentifiers/Entity	The identification number of the care provider

boxIdentifiers/entityType	The type of identification number of the care provider
boxIdentifiers/Quality	The quality of the care provider
currentSize	The current size of the eHealthBox expressed in bytes
notificationEnabled	Whether the box owner opted to receive notifications
unreadMessagesCount	Count of the messages that are unread
standbyMessagesCount	The number of messages that are in standby because the mailbox is full. To consult these messages, the user must delete others. It is necessary to clean the mailbox until the current size is lower than the max size (quota). The global size of an eHealthBox is limited to **10MB** by default (including inbox and bin folder). The messages will then appear in the inbox folder.
actor	Object with first name, last name, SSIN, email and user if the actor is an individual; Or organization name, organization and user if the actor is an organization.
actor/firstName or organizationName	First name or organization name of the box owner
actor/lastName	Last name of the box owner
actor/email	The email of the actor, used to send notifications
actor/ssin	SSIN of the natural person
actor/organization	True if the box owner is an organization.
actor/user	True if the box owner is a natural person.
quota	The maximum size of the mailbox expressed in bytes.
outOfOffices	Object with the active OutOfOffice settings by means of OoOId, startDate, endDate and any substitutes (entity, quality and entityType)

Examples:

natural person:

```
{
   "creationTms": "2019-10-07T15:16:24.848251",
   "lastAccessTms": "2019-10-08T08:27:52.712512",
   "accessKey": {
      "key": "59e2eb3e964ba857578a5c0a41efe482",
      "mailboxIdentifier": {
         "boxIdentifiers": {
            "entity": "79000000000",
            "entityType": "INSS",
            "quality": "DOCTOR"
         }
      }
   },
   "currentSize": 0,
   "notificationEnabled": false,
   "unreadMessagesCount": 0,
   "standbyMessagesCount": 0,
   "actor": {
      "firstName": "Renard",
      "lastName": "Jules",
      "ssin": "79000000000",
      "organization": false,
      "user": true
   },
   "outOfOffices": {
      "1570519822111": {
         "startDate": "2019-10-08",
         "endDate": "2019-10-08",
         "substitutes": []
      },
      "1570521178902": {
         "startDate": "2019-10-09",
         "endDate": "2019-10-09",
         "substitutes": []
      },
      "1570521226299": {
         "startDate": "2019-10-10",
         "endDate": "2019-10-10",
         "substitutes": []
      },
      "1570522540691": {
         "startDate": "2019-10-12",
         "endDate": "2019-10-12",
         "substitutes": [
            {
               "entity": "71000000",
               "quality": "HOSPITAL",
               "entityType": "NIHII"
            }
         ]
      }
   },
   "quota": 10000000
}
```

Legal person:

```
{
   "creationTms": "2019-10-08T08:57:50.502076",
   "lastAccessTms": "2019-10-08T08:57:50.502085",
   "accessKey": {
      "key": "8d9c4c2aa795ffbd11ec79c55fa930aa",
      "mailboxIdentifier": {
         "boxIdentifiers": {
            "entity": "11111111",
            "entityType": "NIHII",
            "quality": "HOSPITAL"
         }
      }
   },
   "currentSize": 0,
   "notificationEnabled": false,
   "unreadMessagesCount": 0,
   "standbyMessagesCount": 0,
   "actor": {
      "organizationName": "HOSPITAL Wilmar 1",
      "organization": true,
      "user": false
   },
   "outOfOffices": {},
   "quota": 10000000
}
```

5.2.2 GET /mailboxes/{accessKey}/folders

Get list of folders for a mailbox

The *getBoxFolders* method allows an authenticated user to receive all folders created in his mailbox.

5.2.2.1 Request

Path parameters

Path parameter	Description
accessKey	The accessKey is a unique key identifier generated in the functionality create mailbox or get accessKey.

Example:

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/59e2eb3e964ba857578a5c0a41efe142/folders
```

5.2.2.2 Response

The response contains a success status code and as many folder elements as there are folders in the considered eHealthBox.

Element	Description
items	0-to-more item tag(s) containing the folder names
total	The total number of folders
value	The folder name which can be one of the following names: ▪ "in" for the inbox folder. ▪ "sent" for the sent box folder.

	▪ "bin" for the messages moved from the inbox folder to the bin. ▪ "binsent" for the messages moved from the sent box folder to the bin.
deletable	Whether the messages in the folder can be deleted (hard)
recoverable	Whether the messages in the folder can be recovered or not. If true, it means the message can be recovered to its initial folder.
trash	Whether the messages in the folder can be moved to a corresponding trash bin. If true, it means the message can be moved to the trash folder.

Example:

```
{
   "items": [
      {
         "value": "in",
         "deletable": true,
         "recoverable": false,
         "trash": true
      },
      {
         "value": "sent",
         "deletable": true,
         "recoverable": false,
         "trash": true
      },
      {
         "value": "bin",
         "deletable": true,
         "recoverable": true,
         "trash": false
      },
      {
         "value": "binsent",
         "deletable": true,
         "recoverable": true,
         "trash": false
      }
   ],
   "total": 4
}
```

5.2.3 GET /mailboxes/{accessKey}/folders/{folder}/messages

Get list of messages

The **getMessageList** method provides a list of messages for a specific folder of your eHealthBox listed in order by date (most recent first, index "1"). A pagination mechanism is available so you can ask for the number of items per page and the page number. Per request, you can send a maximum of one hundred messages.

5.2.3.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey
folder	You need to specify the folder specific to your request via the folder path parameter. The possible values can be retrieved from the *getFolderList* functionality. "in" for the inbox folder. "sent" for the sent box folder. "bin" for the messages moved from the inbox folder to the bin. "binsent" for the messages moved from the sent box folder to the bin.

query parameter	Description
page	The page number is specified via the query parameter, an optional parameter. If not specified the default value will be taken into account. *Default value:* 1
pageSize	The number of messages you want to receive is requested in the *query* parameter, an optional parameter. If not specified the default value will be taken into account. *Default value:* 100 *Maximum value:* 100
hasAnnex	This query parameter allows you to filter on whether the messages have (an) annexe(s. *Default value:* false *True:* only messages with annexes will be in the list *False:* all messages will be shown
important	This query parameter allows you filter on messages sent as important.
messageType	This query parameter allows you filter on the type of message. The following types are available: • *ACKNOWLEDGMENT* • *DOCUMENT* • *ERROR*
q	This query parameter allows you to filter on a string present in the title or the first name, last name, organization name, identifier number (entity) of the sender.
since	This query parameter allows you filter the messages from a specific date. The format has to be yyyy-MM-dd.

Example

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/38d7f36411060620a472545b150d771f/folders/
IN/messages
```

5.2.3.2 Response

The response contains a success status code and as many Message elements as there are messages in the considered folder of the eHealthBox, with maximum the amount of messages as specified in the pageSize parameter.

Each element contains all necessary information to treat the message. There is no difference in content with the getFullMesage (Load a full message). Therefore, each field will be discussed in detail in the getFullMessage chapter.

Element	Description
items	0-to-more item tag(s) describing the eHealthBox Message(s). See load a full message for the detailed description (section 5.2.4);
pageSize	The pageSize contains the number of messages contained in the response;
page	The page number
total	The total Items parameter contains the total number of items in the requested folder.

<u>Examples:</u>

```
{
    "items": [
        {
            "content": {
                "size": 70231,
                "sender": {
                    "actor": {
                        "user": true,
                        "lastName": "CIORSAC",
                        "organization": false
                    },
                    "identifiers": {
                        "entity": "79000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    }
                },
                "annexes": [
                    {
                        "primary": false,
                        "annexKey": "ea765091-e19c-4668-a9f1-c0c6b1fc7ed5",
                        "fileName": "annexname.txt",
                        "contentId": "file"
                    }
                ],
                "original": {
                    "type": "DOCUMENT",
                    "title": "This is a TEST message",
                    "payload": "tNeaOHIfNMnp4",
                    "metadata": {},
                    "encrypted": false,
                    "important": false,
                    "extensions": {
                        "patientNiss": "",
                        "payloadFilename": "Test file",
                        "freeInformations": {
                            "freeText": "Information"
                        }
                    },
                    "recipients": [
                        {
                            "identifiers": {
                                "entity": "90000000000",
                                "quality": "DOCTOR",
                                "entityType": "INSS"
                            },
                            "outOfOfficeIgnored": false
                        }
                    ],
                    "publicationId": "ZN1PHpsb5z",
                    "annexesMetadata": [
                        {
                            "title": "tnjMY3OJv1o8j",
                            "digest": "y3v0dKWwCz5naeP7MOVilxc5C9Qn3Jx33C+0jw3WyJw=",
                            "primary": false,
                            "fileName": "annexname.txt",
                            "contentId": "file",
                            "contentType": "text/plain",
                            "additionalProperties": {}
                        }
                    ],
                    "payloadMimetype": "text/plain",
                    "acknowledgements": {
                        "read": false,
                        "sent": false,
                        "viewed": false
                    }
                },
                "recipient": {
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                },
                "identifier": 3000002847548,
                "expirationDate": "2022-12-14",
                "expirationBinDate": "2022-07-15",
                "expirationSentDate": "2022-07-15",
                "publicationDateTime": "2022-06-15T18:47:48.849655",
                "expirationBinsentDate": "2022-07-15",
                "expirationStandbyDate": "2022-12-14",
            },
            "metadata": {
                "readDateTime": "2019-10-18T10:36:44.345822",
                "viewDateTime": "2019-10-18T09:46:35.247239"
            }
        },
        {
            "content": {
                "size": 335,
                "sender": {
                    "actor": {
                        "user": false,
                        "organization": true,
                        "organizationName": "Noreply"
                    },
                    "identifiers": {
                        "entity": "12345678912",
                        "quality": "CITIZEN",
                        "entityType": "INSS"
                    }
                },
                "annexes": [],
                "original": {
                    "type": "ACKNOWLEDGMENT",
                    "title": "SENT: TestAnnex",
                    "payload": "<!DOCTYPE html><meta http-equiv=\"Content-
Type\" content=\"text/html;charset=UTF-
8\"><html><body>Votre message : (TestAnnex) a été publié dans le système à 11:55:06, 18/10/2019.<br/><b
r/> -------------------------------
- <br/><br/>Uw bericht: (TestAnnex) werd gepubliceerd in het systeem op 11:55:06, 18/10/2019.<\/body><\
/html>",
                    "metadata": {},
                    "encrypted": false,
                    "important": false,
                    "extensions": {
                        "ackType": "SENT",
                        "applicationName": "eHboxSystem",
                        "payloadFilename": "message.html",
                        "originalMessageId": 3000002876557,
                        "originalRecipient": {
```

```
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Nobody",
                        "firstName": "John"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": true
                },
                "originalRecipientAccessKey": "38c7f36411060620a112547a230d891f"
            },
            "recipients": [
                {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Everybody",
                        "firstName": "Jason"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                }
            ],
            "publicationId": "1571392506711",
            "annexesMetadata": [],
            "payloadMimetype": "text/html",
            "acknowledgements": {
                "read": false,
                "sent": false,
                "viewed": false
            }
        },
        "recipient": {
            "person": {
                "ssin": "90000000000",
                "lastName": "Everybody",
                "firstName": "Jason"
            },
            "identifiers": {
                "entity": "90000000000",
                "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                },
                "identifier": 3000002876558,
                "expirationDate": "2022-12-14",
                "expirationBinDate": "2022-07-15",
                "expirationSentDate": "2022-07-15",
                "publicationDateTime": "2022-06-15T18:47:48.849655",
                "expirationBinsentDate": "2022-07-15",
                "expirationStandbyDate": "2022-12-14",
            },
            "metadata": {
                "viewDateTime": "2019-10-18T12:10:00.240902"
            }
        },
        {
            "content": {
                "size": 1849,
                "sender": {
                    "actor": {
                        "user": false,
                        "organization": true,
                        "organizationName": "Noreply"
                    },
                    "identifiers": {
                        "entity": "12345678912",
                        "quality": "CITIZEN",
                        "entityType": "INSS"
                    }
                },
                "annexes": [],
                "original": {
                    "type": "ERROR",
                    "title": "Delivery Status Notification (Failure)",
                    "payload": "<!DOCTYPE html><meta http-equiv=\"Content-
Type\" content=\"text/html;charset=UTF-
8\"><html><body><h2>L'envoi du document suivant a échoué :<\/h2><br/><br/><b>Titre du document : <\/b>K
8WUO4J48MR59<br/><br/><b>Date de publication : <\/b>10:58:15, 18/10/2019<br/><br/><b>Détails techniques
 : <\/b>Les destinataires suivants ne sont pas reconnus par le système. Veuillez vérifier si l'ID du de
stinataire est correct et si le destinataire possède la qualité spécifiée.<br/><br/>L'ID est le numéro
d'identification unique d'une personne ou d'une organisation. Ce numéro peut être le numéro NISS (numér
o de 11 chiffres qui se trouve dans le coin supérieur droit de la carte SIS), le numéro NIHII (numéro I
NAMI de 8 chiffres) ou le numéro CBE (numéro d'entreprise de 10 chiffres).<br/><b>Destinataire(s) : <\/
b><br/>90000000000(INSS) - Dentiste<br/><b>Pièce(s) jointe(s):<\/b><br/><br/>Titre : QW5uZXh0aXRsZQ==<b
r/>Nom du fichier : pdf.pdf<br/><br/><br/><br /><br /> -------------------------------
- <br /><br /><h2>Het verzenden van het volgende document is mislukt :<\/h2><br/><br/><b>Titel van het
```

```
document : <\/b>K8WUO4J48MR59<br/><br/><b>Publicatiedatum : <\/b>10:58:15, 18/10/2019<br/><br/><b>Techn
ische gegevens : <\/b>De volgende ontvangers worden niet herkend door het systeem. Controleer of de ID
van de ontvanger correct is en of hij beschikt over de opgegeven bevoegdheid.<br/><br/>De ID is het uni
eke identificatienummer van een persoon of van een organisatie. Dit nummer kan het INSZ-
nummer (nummer van 11 cijfers op de rechterbovenhoek van de SIS-kaart), het NIHII-nummer (RIZIV-
nummer van 8 cijfers) of het KBO-
nummer (ondernemingsnummer van 10 cijfers) zijn.<br/><b>Ontvanger(s) : <\/b><br/>90000000000(INSS) - Ta
ndarts<br/><b>Bijlagen :<\/b><br/><br/>Titel : QW5uZXh0aXRsZQ==<br/>Bestandsnaam  : pdf.pdf<br/><br/><b
r/><\/body><\/html>",
                    "metadata": {
                        "code": "703",
                        "message": "One or more recipients are invalid.",
                        "originalPublicationId": "97F2VD9C50RJ5"
                    },
                    "encrypted": false,
                    "important": false,
                    "extensions": {
                        "applicationName": "eHboxSystem",
                        "payloadFilename": "message.html",
                        "undeliveredRecipients": [
                            {
                                "identifiers": {
                                    "entity": "90000000000",
                                    "quality": "DENTIST",
                                    "entityType": "INSS"
                                },
                                "outOfOfficeIgnored": false
                            }
                        ]
                    },
                    "recipients": [
                        {
                            "person": {
                                "ssin": "90000000000",
                                "lastName": "Everybody",
                                "firstName": "Jason"
                            },
                            "identifiers": {
                                "entity": "90000000000",
                                "quality": "DOCTOR",
                                "entityType": "INSS"
                            },
                            "outOfOfficeIgnored": false
                        }
                    ],
                    "publicationId": "1571389102801",
                    "annexesMetadata": [],
                    "payloadMimetype": "text/html",
                    "acknowledgements": {
                        "read": false,
                        "sent": false,
                        "viewed": false
                    }
                },
                "recipient": {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Everybody",
                        "firstName": "Jason"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                },
                "identifier": 3000002876553,
                 "expirationDate": "2022-12-14",
                "expirationBinDate": "2022-07-15",
                "expirationSentDate": "2022-07-15",
                "publicationDateTime": "2022-06-15T18:47:48.849655",
                "expirationBinsentDate": "2022-07-15",
                "expirationStandbyDate": "2022-12-14",
            },
            "metadata": {
                "viewDateTime": "2019-10-18T11:49:25.520882"
            }
        }
    ],
    "page": 1,
    "pageSize": 3,
    "total": 3
}
```

5.2.3.3 *Remarks*

- The viewDateTime will be set for the messages viewed for the first time. This is only valid for the messages in the IN and the BIN.
- If acknowledgement viewed is set to true, a view acknowledgement is sent to the sender by the system box.
- The folder can only contain the possible values as mentioned in *getBoxFolders* functionality.
- The page parameter is a numeric value and larger than 0.
- The pageSize parameter is a numeric value between 0 and 100.

5.2.4 GET /mailboxes/{accessKey}/folders/{folder}/messages/{messageId}

The *getFullMessage* method is used to get the corresponding complete message and its content to a provided MessageId. The messageId can be retrieved through a getMessageList (chapter 5.2.3) and can be found in element identifier.

5.2.4.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get accesskey
Folder	You need to specify the folder for your request via the folder path parameter. The possible values can be retrieved from the *getFolderList* functionality. "in" for the inbox folder. "sent" for the sent box folder. "bin" for the messages moved from the inbox folder to the bin. "binsent" for the messages moved from the sent box folder to the bin.
messageId	The mandatory messageId is a unique message-IDentification generated by the system. The messageId is retrieved when calling upon the *getMessagesList* under identifier element

5.2.4.2 Response

Element	Description
Content	Object with the message inside
Size	Size of the message in bytes
Sender	Object with the sender information: contains object actor and identifiers.
Actor	Object with SSIN, user, last name, first name and organization key
actor/ssin	National security number of the sender, also contains the SSIN of a natural person sending a message on behalf of an organization
actor/user	True = sender is a natural person. If the care provider has sent the mail on behalf of an organization, it will be false
actor/lastName	Last name of the sender
actor/firstName or organizationName	First name or/and Organization name of the sender. If the care provider has sent the mail on behalf of an organization, the actor object will have organizationName filled in.
actor/organization	True= sender is an organization, false= sender is a natural person.
identifiers	Object with the sender box references: Entity EntityType Quality If the care provider has sent the mail on behalf of an organization, these must be the references of the organization.

identifiers/entity	The identification number of the sender is provided.
identifiers/entityType	The type of identification number of the sender is provided.
identifiers/quality	The quality of the sender is provided
annexes	Array with the annex information for each annex
annexes/primary	Whether the annex was put in the body (true) or as annex (false). In eHBox REST – v1 (**previously called V4**) it is no longer possible to put an annex in the body.
annexes/annexKey	Unique key of the annex, used to load the attachment
annexes/fileName	File name of the annex
Annexes/contentId	Unique alphanumerical value per message for each annex
original	Object with the original immutable received message
type	DOCUMENT/ERROR/ACKNOWLEDGMENT
title	Title of the message
payload	The payload of the message.
metadata	Object metadata. Currently, no meta information is defined. Additional system meta information can be defined by the eHealth platform and used in agreement with the client (for future needs). The type of meta information must be defined in the eHealthBox system before it can be used.
encrypted	BOOLEAN (true or false) that indicates if the content has been encrypted.
important	BOOLEAN (true or false) that indicates if the message is to be considered as important.
extensions	The sender is free to add more information via extensions provided transparently to the recipient(s). The following extensions are interpreted by the SOAP web service: payloadFilename applicationName freeInformations ehealthMeta patientNiss Extensions other than listed above will be shown in the response, but will not be interpreted by eHealth or shown in a SOAP call. The following extensions are automatically generated for an acknowledgement: ackType applicationName payloadFilename originalMessageId originalRecipient originalRecipientAccessKey The following extensions are automatically generated for an error: applicationName

	payloadFilename undeliveredRecipients : array with undelivered recipient details
extensions/payloadFilename	The name of the attachment sent as payload.
extensions/applicationName	The Application sending the message
extensions/freeInformations	Object where free info was added by the sender. The following keys will be interpreted by eHealthBox: Free Text (encryptable) Table with a title and rows Rows: left cell and right cell (encryptable)
Extensions/ehealthMeta	Array of values. Additional system meta information can be defined by the eHealth platform and used in agreement with the client (for future needs). The type of meta information must be defined in the eHealthBox system before it can be used.
extensions/patientNiss	This optional field allows to specify an SSIN number of a patient concerned by the message content.
extensions/acktype	Can be "READ", "PUBLISHED" or "RECEIVED"
extensions/originalMessageId	Original messageId the acknowledgement is about.
extensions/originalRecipient	The box identifiers of the message recipient regarding the acknowledgement. It contains the following: • the person object with SSIN, last name and first name • identifiers object with entity, quality and entityType of the recipient • outOfOfficeIgnored BOOLEAN of the recipient • originalRecipientAccessKey: the recipient unique technical key
recipients	Array with the recipients of the message. Each recipient can have a person object, but will always have an identifiers object and an outOfOfficeIgnored Each recipient is defined by: • (person): - firstName (of the recipient) - lastName (of the recipient) - SSIN (of the recipient) • identifiers (see definition above in this table) • outOfOfficeIgnored: True = the recipient will receive the message regardless of an active out of office. False: the sender will receive an error message in case of an active out of office.
sentAsNihii	If the message was sent using the NIHII of a natural person instead of his SSIN.
publicationId	publication ID of the message
annexesMetadata	Array with metadata for each annex

	Each object contains: • title • digest • primary: true if annex comes from the payload • fileName • contentId • contentType: mimetype of the annex • additionalProperties: free to use metadata with key/value pairs
payloadMimetype	Mimetype of the payload. For messages generated in REST, it will always be text/plain or text/html
acknowledgements	Object with following key-value pairs: • read (BOOLEAN) • sent (BOOLEAN) • viewed (BOOLEAN)
recipient	Object with the information about the current recipient, the same information is repeated from recipients, but this time only for the current recipient.
identifier	Unique identifier of the message
expirationDate	Date of expiration of a message in the folder “in”
expirationBinDate	Date of expiration of the message in the folder “bin”
expirationSentDate	Date of expiration of the message in the folder “sent”
publicationDateTime	Date of publication
expirationBinsentDate	Date of expiration of the message in the folder “binsent”
expirationStandbyDate	Date of expiration of the message in the folder “standby”
metadata	Contains the readDateTime and viewDateTime of the message if the message is situated in the "bin" or "in" folder.

Examples:

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/59e2eb3e964ba857578a5c0a41efe142/folders/
IN/messages/3100000952364
```

Document:

```
{
    "content": {
        "size": 70231,
        "sender": {
            "actor": {
                "user": true,
                "lastName": "CIORSAC",
                "organization": false
            },
            "identifiers": {
                "entity": "79000000000",
                "quality": "DOCTOR",
                "entityType": "INSS"
            }
        },
        "annexes": [
            {
                "primary": false,
                "annexKey": "ea765091-e19c-4668-a9f1-c0c6b1fc7ed5",
                "fileName": "annexname.txt",
                "contentId": "1"
            }
        ],
        "original": {
            "type": "DOCUMENT",
            "title": "This is a TEST message",
            "payload": "tNeaOHIfNMnp4",
            "metadata": {},
            "encrypted": false,
            "important": false,
            "extensions": {
                "patientNiss": "",
                "payloadFilename": "Test file",
                "freeInformations": {
                    "freeText": "Information"
                }
            },
            "recipients": [
                {
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                }
            ],
            "publicationId": "ZN1PHpsb5z",
            "annexesMetadata": [
                {
                    "title": "tnjMY3OJv1o8j",
                    "digest": "y3v0dKWwCz5naeP7MOVilxc5C9Qn3Jx33C+0jw3WyJw=",
                    "primary": false,
                    "fileName": "annexname.txt",
                    "contentId": "1",
                    "contentType": "text/plain",
                    "additionalProperties": {}
                }
            ],
            "payloadMimetype": "text/plain",
            "acknowledgements": {
                "read": false,
                "sent": false,
                "viewed": false
            }
        },
        "recipient": {
            "identifiers": {
                "entity": "90000000000",
                "quality": "DOCTOR",
                "entityType": "INSS"
            },
            "outOfOfficeIgnored": false
        },
        "identifier": 3000002847548,
        "expirationDate": "2022-12-14",
        "expirationBinDate": "2022-07-15",
        "expirationSentDate": "2022-07-15",
        "publicationDateTime": "2022-06-15T18:47:48.849655",
        "expirationBinsentDate": "2022-07-15",
        "expirationStandbyDate": "2022-12-14",


    },
    "metadata": {
        "readDateTime": "2019-10-18T10:36:44.345822",
        "viewDateTime": "2019-10-18T09:46:35.247239"
    }
}
```

Acknowledgment:

```
{
    "content": {
        "size": 335,
        "sender": {
            "actor": {
                "user": false,
                "organization": true,
                "organizationName": "Noreply"
            },
            "identifiers": {
                "entity": "12345678912",
                "quality": "CITIZEN",
                "entityType": "INSS"
            }
        },
        "annexes": [],
        "original": {
            "type": "ACKNOWLEDGMENT",
            "title": "SENT: TestAnnex",
            "payload": "<!DOCTYPE html><meta http-equiv=\"Content-
Type\" content=\"text/html;charset=UTF-
8\"><html><body>Votre message : (TestAnnex) a été publié dans le système à 11:
55:06, 18/10/2019.<br/><br/> -------------------------------
- <br/><br/>Uw bericht: (TestAnnex) werd gepubliceerd in het systeem op 11:55:
06, 18/10/2019.<\/body><\/html>",
            "metadata": {},
            "encrypted": false,
            "important": false,
            "extensions": {
                "ackType": "SENT",
                "applicationName": "eHboxSystem",
                "payloadFilename": "message.html",
                "originalMessageId": 3000002876557,
                "originalRecipient": {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Nobody",
                        "firstName": "John"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": true
                },
                "originalRecipientAccessKey": "38c7f36411060620a112547a230d891
f"
            },
            "recipients": [
                {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Everybody",
                        "firstName": "Jason"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                }
            ],
            "publicationId": "1571392506711",
            "annexesMetadata": [],
            "payloadMimetype": "text/html",
            "acknowledgements": {
                "read": false,
                "sent": false,
                "viewed": false
            }
        },
        "recipient": {
            "person": {
                "ssin": "90000000000",
                "lastName": "Everybody",
                "firstName": "Jason"
            },
            "identifiers": {
                "entity": "90000000000",
                "quality": "DOCTOR",
                "entityType": "INSS"
            },
            "outOfOfficeIgnored": false
        },
        "identifier": 3000002876558,
        "expirationDate": "2022-07-15",
        "expirationBinDate": "2022-07-15",
        "expirationSentDate": "2022-07-15",
        "publicationDateTime": "2022-06-15T18:47:48.849655",
        "expirationBinsentDate": "2022-07-15",
        "expirationStandbyDate": "2022-07-15",
    },
    "metadata": {
        "viewDateTime": "2019-10-18T12:10:00.240902"
    }
}
```

Error:

```
{
    "content": {
        "size": 1849,
        "sender": {
            "actor": {
                "user": false,
                "organization": true,
                "organizationName": "Noreply"
            },
            "identifiers": {
                "entity": "12345678912",
                "quality": "CITIZEN",
                "entityType": "INSS"
            }
        },
        "annexes": [],
        "original": {
            "type": "ERROR",
            "title": "Delivery Status Notification (Failure)",
            "payload": "<!DOCTYPE html><meta http-equiv=\"Content-
Type\" content=\"text/html;charset=UTF-
8\"><html><body><h2>L'envoi du document suivant a échoué :<\/h2><br/><br/><b>T
itre du document : <\/b>K8WUO4J48MR59<br/><br/><b>Date de publication : <\/b>1
0:58:15, 18/10/2019<br/><br/><b>Détails techniques : <\/b>Les destinataires su
ivants ne sont pas reconnus par le système. Veuillez vérifier si l'ID du desti
nataire est correct et si le destinataire possède la qualité spécifiée.<br/><b
r/>L'ID est le numéro d'identification unique d'une personne ou d'une organisa
tion. Ce numéro peut être le numéro NISS (numéro de 11 chiffres qui se trouve 
dans le coin supérieur droit de la carte SIS), le numéro NIHII (numéro INAMI d
e 8 chiffres) ou le numéro CBE (numéro d'entreprise de 10 chiffres).<br/><b>De
stinataire(s) : <\/b><br/>90000000000(INSS) - Dentiste<br/><b>Pièce(s) jointe(
s):<\/b><br/><br/>Titre : QW5uZXh0aXRsZQ==<br/>Nom du fichier : pdf.pdf<br/><b
r/><br/><br /><br /> -------------------------------
- <br /><br /><h2>Het verzenden van het volgende document is mislukt :<\/h2><b
r/><br/><b>Titel van het document : <\/b>K8WUO4J48MR59<br/><br/><b>Publicatied
atum : <\/b>10:58:15, 18/10/2019<br/><br/><b>Technische gegevens : <\/b>De vol
gende ontvangers worden niet herkend door het systeem. Controleer of de ID van
 de ontvanger correct is en of hij beschikt over de opgegeven bevoegdheid.<br/
><br/>De ID is het unieke identificatienummer van een persoon of van een organ
isatie. Dit nummer kan het INSZ-
nummer (nummer van 11 cijfers op de rechterbovenhoek van de SIS-
kaart), het NIHII-nummer (RIZIV-nummer van 8 cijfers) of het KBO-
nummer (ondernemingsnummer van 10 cijfers) zijn.<br/><b>Ontvanger(s) : <\/b><b
r/>90000000000(INSS) - Tandarts<br/><b>Bijlagen :<\/b><br/><br/>Titel : QW5uZX
h0aXRsZQ==<br/>Bestandsnaam  : pdf.pdf<br/><br/><br/><\/body><\/html>",
            "metadata": {
                "code": "703",
                "message": "One or more recipients are invalid.",
                "originalPublicationId": "97F2VD9C50RJ5"
            },
            "encrypted": false,
            "important": false,
            "extensions": {
                "applicationName": "eHboxSystem",
                "payloadFilename": "message.html",
                "undeliveredRecipients": [
                    {
                        "identifiers": {
                            "entity": "90000000000",
                            "quality": "DENTIST",
                            "entityType": "INSS"
                        },
                        "outOfOfficeIgnored": false
                    }
                ]
            },
            "recipients": [
                {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Everybody",
                        "firstName": "Jason"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                }
            ],
            "publicationId": "1571389102801",
            "annexesMetadata": [],
            "payloadMimetype": "text/html",
            "acknowledgements": {
                "read": false,
                "sent": false,
                "viewed": false
            }
        },
        "recipient": {
            "person": {
                "ssin": "90000000000",
                "lastName": "Everybody",
                "firstName": "Jason"
            },
            "identifiers": {
                "entity": "90000000000",
                "quality": "DOCTOR",
                "entityType": "INSS"
            },
            "outOfOfficeIgnored": false
        },
        "identifier": 3000002876553,
        "expirationDate": "2022-07-15",
        "expirationBinDate": "2022-07-15",
        "expirationSentDate": "2022-07-15",
        "publicationDateTime": "2022-06-15T18:47:48.849655",
        "expirationBinsentDate": "2022-07-15",
        "expirationStandbyDate": "2022-07-15",
    },
    "metadata": {
        "viewDateTime": "2019-10-18T11:49:25.520882"
    }
}
```

5.2.4.3 Remarks

- The readDateTime will be set for the message when it is read for the first time. This is only valid for the messages in the IN and the BIN.
- If acknowledgement read was set to true, the system box will send a read acknowledgement to the sender.
- The folder can only contain the possible values as mentioned in getBoxFolders functionality.
- The messageId needs to be available in the specified mailbox, within the specified mailbox folder. Valid messageIds can be retrieved from the message list (getMessageList method).

5.2.5 GET/mailboxes/{accessKey}/folders/{folder}/messages/{messageId}/attachments/{attachmentKey}

Load an attachment by id

The getAttachment method is used to get the corresponding attachment to a provided attachmentKey.

5.2.5.1 Request

Path parameters

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey.
folder	You need to indicate the folder specific to your request via the folder path parameter. The possible values can be retrieved from the getFolderList functionality. • "in" for the inbox folder. • "sent" for the sent box folder.

	It is not possible to download an attachment from the bin.
messageId	The mandatory messageId is a unique message identification generated by the system. The messageId is retrieved when calling upon the getMessagesList.
attachmentKey	The mandatory attachmentKey is a unique key identification generated by the system and referenced to in the getFullMessage method.

Example

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/59e2eb3e900ba850000a5c0a41efe482/folders/
IN/messages/3000000000550/attachments/67f1130b-c606-4814-255e-739848e7bb60
```

5.2.5.2 *Response*

The response is a download of the attachment.

Element	Description
messageId	The Id generated for the published message
publicationId	When a publicationId was specified, the API will return it in the response.
href	Link to get the status of the acknowledgement

5.2.5.3 *Remarks*

- The folder can only contain the possible values as mentioned in *getFolderList* functionality.
- The messageId needs to be available in the specified mailbox, within the specified mailbox folder.
- The attachmentKey needs to be a valid key, the key can be retrieved in the body of a message with attachments.

5.2.6 GET /mailboxes/{accessKey}/publications/{messageId}

The GET acknowledgement method renders the acknowledgement information about a specific message.

5.2.6.1 *Request*

Path parameters

Path parameter	Description
accessKey	The accessKey is a unique key identifier generated in the functionality create mailbox or get accessKey.
messageId	The mandatory messageId is a unique message identification generated by the system. The messageId is retrieved when calling upon the *getMessagesList*. String of 13 digits.

```
GET https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/38c7f36411060620a112547a230d891f/publicat
ions/3000000000798
```

5.2.6.2 Response

The response contains a success status code or a Business Error.

Element	Description
items	Array with the recipients with the acknowledgement status information • recipient • publishDateTime • viewDateTime • readDateTime
total	Number of recipients of the message

Example:

```
{
    "items": [
        {
            "recipient": {
                "identifiers": {
                    "entity": "79000000000",
                    "quality": "DOCTOR",
                    "entityType": "INSS"
                },
                "outOfOfficeIgnored": true
            },
            "publishDateTime": "2020-10-14T16:12:48.917101",
            "viewDateTime": "2020-10-14T16:26:02.252956",
            "readDateTime": "2020-10-14T16:27:38.28232"
        },
        {
            "recipient": {
                "identifiers": {
                    "entity": "79101228913",
                    "quality": "DOCTOR",
                    "entityType": "INSS"
                },
                "outOfOfficeIgnored": false
            },
            "publishDateTime": "2020-10-14T16:12:48.917101"
        }
    ],
    "total": 2
}
```

5.3 Publication

5.3.1 POST /mailboxes/{accessKey}/publications

The publishMessage method enables the user to publish a new message to the mailbox of another user.

The publishMessage method is a POST of type multipart/form-data. Each part must be a file.

The structure is the following:

- JSON message as a file, identified as 'body' key and content-type: application/json
- 0 to many annexes, identified with a unique per request alphanumerical key (contentId) and content-type: the mimetype of the annex

Each annex should be present in the annexesMetadata array with the matching contentId.

A java example for building a message with an annex can be found in Chapter 8 - Annexes.

5.3.1.1 Request

Path parameters

Path parameter	Description
accessKey	The accessKey is a unique key identifier of the user of eHealthBox generated in the functionality create mailbox or get accessKey.

Body

Element	Description	encryptable	mandatory
type	DOCUMENT (only allowed type for publication)		yes
publicationId	Identifier of the publication that must unique per sent box. If it is not unique, it will asynchronously result in a delivery failure. This field is depreciated and it is recommended to stop using it.		no
title	Title of the message		yes
recipients	Array with the recipients of the message. Each recipient can have a person object, but will always have an identifiers object and an outOfOfficeIgnored Each recipient is defined by: • (person) • identifiers • outOfOfficeIgnored		yes
person/firstName or OrganizationName	First name or Organization name of the recipient		mandatory for individual
person/lastName	Last name of the recipient		mandatory for individual

person/ssin	National security number of the recipient		mandatory for individual
identifiers	Object with the sender box references: • entity • entityType • quality		yes
identifiers/entity	The identification number of the recipient		yes
Identifiers/entityType	The type of identification number of the recipient		yes
identifiers/quality	The quality of the recipient		yes
outOfOfficeIgnored	If set to true, the Out of Office of the recipient will be ignored		yes
payload	The payload of the message. Its value represents the textual body of the message	X	yes
payloadMimetype	Mime type of the payload. Only text/html or text/plain are allowed as mime type for the payload		yes
acknowledgements	Object with following key-value pairs: • read • sent • viewed		yes
acknowledgements/read	If true, an acknowledgement is sent when the recipient executes a getFullMessage request		no (default = true)
acknowledgements/sent	If true, an acknowledgement is sent when the message remains in the recipient inbox		no (default = true)
acknowledgements/viewed	If true, an acknowledgement is sent when the recipient executes a getMessageList request		no (default = true)
encrypted	If true, all encryptable fields have to be in base 64 with padding		no (default = false)
important	If true, the message is sent as important		no (default = false)
metadata	Additional system meta information		no
annexesMetadata	Array with metadata for each annex Each object contains: • title (encryptable and mandatory) • filename (mandatory) • digest (SHA-256 encoded in B64) (optional)	X	yes, for each annex

	• contentType: mimetype of the annex (optional) • additionalProperties: object where the sender can add additional properties to the annex • contentId: Unique alphanumerical value per message for each annex and that corresponds with the value defined in the multipart		
extensions	The sender is free to add more information via extensions, provided transparently to the recipient(s). The following extensions are interpreted by the SOAP webservice: • freeInformations • ehealthMeta • patientNiss Extensions other than listed above will be shown in the response, but will not be interpreted by eHealth or shown in a SOAP call.	X (only the fields that are interpreted by eHealth)	no
freeInformations	Object where free info was added by the sender. The following keys are mutual exclusive and will be interpreted by eHealth: • freeText (encryptable) • table with a title and rows • rows: left cell and right cell (encryptable)	X (only the fields that are interpreted by eHealth	no

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/59e2eb3e964ba857578a5c0a41efe142/publicat
ions
{
    "type": "DOCUMENT",
    "publicationId": "LJ3GAOELKZ33K",
    "title": "TestMessage",
    "recipients": [
        {
            "person": {
                "firstName": "John",
                "lastName": "Nobody",
                "ssin": "90000000000"
            },
            "identifiers": {
                "entity": "90000000000",
                "entityType": "INSS",
                "quality": "DOCTOR"
            },
            "outOfOfficeIgnored": false
        }
    ],
    "payload": "This is a test message",
    "payloadMimetype": "text/plain",
    "acknowledgements": {
        "sent": true,
        "read": true,
        "viewed": true
    },
    "encrypted": false,
    "important": true,
    "metadata": {
        "meta1": "value1",
        "meta2": "value2"
    },
    "extensions": {
        "patientNiss": "79000000000",
        "freeInformations": {
            "freeText": "ABCDABCD"
        }
    }
}
```

5.3.1.2 Response

Element	Description
messageId	The generated Id for the published message
publicationId	If a publicationId was specified, the API will return it in the response.
href	Link to get the status of the acknowledgement

example

```
{
    "messageId" : 3100000970020,
    "publicationId" : "T9RPVEONH79PC",
    "href" : "/ehBox/mailboxes/38c7f36411060620a112547a230d891f/publications/3
100000970020"
 }
```

5.3.1.3 Remarks

- If flag encrypted is true, the payload + extensions, patientNiss (if present) + any annexMetadata title must be encoded in Base64
- The payloadMimetype must be either "text/html" or "text/plain"
- The message type must be 'DOCUMENT'
- Any key/value metadata cannot have an empty key or empty value.

5.4 Moving and Deleting

5.4.1 POST /mailboxes/{accessKey}/folders/in/messages/trash

Post a request to trash messages from inbox to binbox

The trashINMessage method enables the user to move a message from source inbox to destination binbox.

5.4.1.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey.
ids	JSON array with the messageIds that have to be moved. This Id can be found in the getMessageList or getFullMessage method under the identifier element or in the response of an publication under messageId.

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/540540/folders/in/messages/trash

{
    "ids": [
        "125",
        "126"
    ]
}
```

5.4.1.2 Response

The response contains a success status code or a Business Error.

The invalid message-IDs or not found in the "in" mailbox folder will be returned in the response and the http code will be 200.

If all messages were successfully moved, the http code will be 204.

Element	Description
items	Array with the message-ID's that could not be moved.
total	Number of messages that could not be moved.

Example

```
{
    "items": [
        125,
        126
    ],
    "total": 2
}
```

5.4.2 POST /mailboxes/{accessKey}/folders/bin/messages/recover

Post a recovery on messages to recover them to their inbox

This recoverINMessage method enables the user to move a message from source binbox to destination inbox.

5.4.2.1 *Request*

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey;
ids	JSON array with the messageIds that have to be moved. The id of a message can be found in the getMessageList or getFullMessage method under the identifier element or in the response of an publication under messageId.

In the body of the request:

Body : {"ids": [<integer>]}

In 'ids' field, specify as many elements as there are messages to be moved.

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/540540/folders/bin/messages/recover

{
    "ids": [
        125,
        126
    ]
}
```

5.4.2.2 *Response*

The response contains a success status code or a Business Error.

The invalid message-IDs or not found in the "bin" mailbox folder will be returned in the response.

Element	Description
items	Array with the message-ID's that could not be moved.
total	Number of messages that could not be moved.

Example:

```
{
    "items": [
        125,
        126
    ],
    "total": 2
}
```

5.4.3 POST /mailboxes/{accessKey}/folders/sent/messages/trash

Post a request to trash messages from sentbox to binsentbox

This trashSENTMessage method enables the user to move a message from source sentbox to destination binsentbox.

5.4.3.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey
ids	JSON array with the messageIds that have to be moved.

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/540540/folders/sent/messages/trash
{
    "ids": [
        "125",
        "126"
    ]
}
```

5.4.3.2 Response

The response contains a success status code or a Business Error.

The invalid message-IDs or not found within the "in" mailbox folder will be returned in the response.

Element	Description
items	array with the message-ID's that could not be moved
total	number of messages that could not be moved

Example

```
{
    "items": [
        125,
        126
    ],
```

5.4.4 POST /mailboxes/{accessKey}/folders/binsent/messages/recover

Post a recovery on messages to restore them to their sent box

This recoverSENTMessage method enables the user to move a message from source binsentbox to destination sent box.

5.4.4.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey;
ids	JSON array with the message-IDs that have to be moved. The Id can be found in the getMessageList or getFullMessage method under the identifier element or in the response of an publication under messageId;

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/540540/folders/binsent/messages/recover
 {
    "ids": [
        125,
        126
    ]
}
```

5.4.4.2 Response

The response contains a success status code or a Business Error.

The invalid message-IDs or not found in the "binsent" mailbox folder will be returned in the response.

Element	Description
items	Array with the message-ID's that could not be moved
total	Number of messages that could not be moved

Example

```
{
    "items": [
        125,
        126
    ],
    "total": 2
}
```

5.4.5 DELETE /mailboxes/{accessKey}/folders/{folder}/messages/{messageId}

Delete a message

The delete a message method enables the user to delete one specific message.

5.4.5.1 Request

Path parameters

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey.
folder	You need to indicate the folder specific to your request via the folder path parameter. The possible values can be retrieved from the *getFolderList* functionality. • "in" for the inbox folder. • "sent" for the sent box folder. • "bin" for the messages moved from the inbox folder to the bin. • "binsent" for the messages moved from the sent box folder to the bin.
messageId	The message-ID the user wants to delete. The Id of a message can be found in the getMessageList or getFullMessage method under the identifier element or in the response of an publication under messageId.

5.4.5.2 Response

If the authentication, folder and accessKey are correct, the response will always be an https code 204 indicating whether the messageId exists.

5.4.6 POST /mailboxes/{accessKey}/folders/{folder}/messages/delete

Post a request to delete multiple messages

This deleteMessages method enables the user to delete one or more messages from the mailbox.

5.4.6.1 Request

Path parameters

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey.
folder	You need to indicate the folder specific to your request via the folder path parameter. The possible values can be retrieved from the *getFolderList* functionality. • "in" for the inbox folder. • "sent" for the sent box folder. • "bin" for the messages moved from the inbox folder to the bin. • "binsent" for the messages moved from the sent box folder to the bin.
ids	The message-ID's that the user wants to delete. The id of a message can be found in the getMessageList or getFullMessage method under the identifier element or in the response of an publication under messageId.

Example

```
POST https://api-
acpt.ehealth.fgov.be/ehBox/mailboxes/540540/folders/in/messages/delete

{
    "ids": [
        125,
        126
    ]
}
```

5.4.6.2 *Response*

The response contains a success status code or a Business Error.

The invalid message-IDs or not found in the mailbox folder will be returned in the response.

Element	Description
items	Array with the message-ID's that could not be deleted
total	Number of messages that could not be deleted

Example

```
{
    "items": [
        125,
        126
    ],
    "total": 2
}
```

5.5 Out of Office

5.5.1 POST /mailboxes/{accessKey}/outOfOffices

Create an OutOfOffice for a specific mailbox

This insertOoO method enables the user to insert a new Out-of-Office configuration for a specified Mailbox.

5.5.1.1 *Request*

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey.

jsonbody	Description
startDate	Date the Out-Of-Office initiates, specified date included

endDate	Date the Out-Of-Office ends, specified date included
substitutes	Array of 0 to 5 substitutes (specified by ID, Type, quality)

example

```
POST https://api-acpt.ehealth.fgov.be/ehBox/mailboxes/540540/outOfOffices

{
    "startDate": "2019-10-28",
    "endDate": "2019-10-28",
    "substitutes": [
        {
            "entity": "80000000000",
            "entityType": "INSS",
            "quality": "DOCTOR"
        }
    ]
}
```

5.5.1.2 *Response*

The response contains a success status code or a Business Error. Eventual substitutes that were not accepted will be listed in the response.

Element	Description
outOfOfficeId	The unique Out-of-Office identifier is automatically generated by the system and can be used as reference for deletion. It will only be present when the out of office has successfully been added.
success	The success field identifies if the Out-of-Office configuration has been created successfully.
substitutesInError	An array with the substitutes that caused the error. Each substitute that causes the problem will have the following object and key: • identifiers (entity, entityType and quality) • linkedErrorCodeValue (one of the business error codes listed in chapter 8.2) • outOfOfficeStartDate: this field represents the out of office start date of the substitute. It will be in the response if for the chosen period the substitute has an active OoO. • outOfOfficeEndDate: this field represents the out of office end date of the substitute. It will be in the response if for the chosen period the substitute has an active OoO.
Instance	Unique identifier used to trace the session of the user

Examples:

success

```
{
    "success": true,
    "outOfOfficeId": "1605686737356",
    "substitutesInError": []
}
```

Error type 1

```
{
    "success": false,
    "substitutesInError": [
        {
            "identifiers": {
                "entity": "90000000000",
                "entityType": "INSS",
                "quality": "DOCTOR"
            },
            "linkedErrorCodeValue": "825"
        },
        {
            "identifiers": {
                "entity": "42000000000",
                "entityType": "INSS",
                "quality": "DOCTOR"
            },
            "linkedErrorCodeValue": "825"
        },
        {
            "identifiers": {
                "entity": "101",
                "entityType": "INSS",
                "quality": "DOCTOR"
            },
            "linkedErrorCodeValue": "827"
        },
        {
            "identifiers": {
                "entity": "79000000000",
                "entityType": "INSS",
                "quality": "DOCTOR"
            },
            "outOfOfficeStartDate": "2020-11-18",
            "outOfOfficeEndDate": "2020-11-25",
            "linkedErrorCodeValue": "824"
        }
    ]
}
```

Error type 2 (see chapter 8.2)

```
{
    "title": "Bad request",
    "detail": "The period 31/12/2019 to 31/12/2019 is invalid because it overl
aps another period.",
    "instance": "bd27761b1fb85bb6",
    "code": "820"
}
```

5.5.1.3 Remarks

- The number of out of offices for one mailbox may not exceed 10.
- The start date is before or the same date as the end date.
- An out-of-office cannot overlap another out-of-office configuration in period.
- The end date cannot be more than a year in the future.
- The start date cannot be in the past.
- A substitute cannot be chosen when he has an active out-of-office configuration in the same period.
- The number of substitutes cannot exceed 5.
- A substitute needs to be known as person (not as organization) by CoBRHA. (urn:be:fgov:person:ssin)
- A person cannot be substitute for himself.
- A substitute should have an active eHealthBox.

5.5.2 DELETE /mailboxes/{accessKey}/outOfOffices/{id}

Delete an Out of Office

5.5.2.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey;
id	The id due for deletion The out-of-office id is a unique identification for an out-of-office configuration generated by the system. The out-of-office id is retrieved when calling the getBoxInfo.

5.5.2.2 Response

The response contains a success status code (http code 204) or a Business Error.

Example of an error:

```
{
    "title": "Not found",
    "detail": "The OutOfOffice 1572516734748 does not exist for the accesskey
a3a2dcbfb5651cc82d2c454b50bb000c",
    "instance": "d5fa89f89e22e1bd",
    "code": "840"
}
```

5.6 Notification settings

5.6.1 PATCH /mailboxes/{accessKey}

The patch mailboxes enables the user to update notification settings of mailbox

5.6.1.1 Request

Path parameter	Description
accessKey	The mandatory accessKey is the unique key identifier of the REST-service user generated in the functionality Create mailbox or get Accesskey

Element	Description
mailBoxPatch	A notification configuration consists of 2 parameters : • Email • notificationEnabled

<u>Example</u>

```
PATCH /mailboxes/540540

{
    "email": "123@test.com",
    "notificationEnabled": true
}
```

5.6.1.2 Response

The response contains a success status code (http code 204) or an error 401, 403, 404.

6. Risks and security

6.1 Security

6.1.1 Business security

In case the development adds a use case based on an existing integration, the eHealth platform must be informed at least one month in advance. A detailed estimate of the expected load is necessary to be able to ensure an effective capacity management.

When technical issues occur on the WS, the partner can obtain support from the contact centre (see Chap 3)

If the eHealth platform should find a bug or vulnerability in its software, the partner must update his application with the latest version of the software, within ten (10) business days.

If the partner finds a bug or vulnerability in the software or web service made available by the eHealth platform, he is obliged to contact and inform us immediately. He is not allowed, under any circumstances, to publish this bug or vulnerability.

6.1.2 The use of username, password and token

The username, password, and token are strictly personal.
Every user takes care of his username, password and token, and he is forced to confidentiality of it. It is prohibited to transfer them to partners and clients. Until inactivation, every user is responsible for every use, including the use by a third party.

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.

In some cases, the eHealth platform provides you, upon request, with test cases in order for you to test your client before releasing 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.

7.2 Error and failure messages Http codes

These are the error status codes that can be returned by the eHealthBox Rest service:

Code	Message	Description
200	OK	The resource has been fetched and is transmitted in the message body or the resource describing the result of the action is transmitted in the message body.
201	Created	Indicates that the request has succeeded and has led to the creation of the resource.
202	Accepted	The publication request has been accepted, but it is still being processed and might deliver a delivery failure.
204	No content	The request has succeeded.
400	Bad request	The structure of the query parameters or the JSON body is not correct.

401	Not authenticated	The user did not sent a valid JWT with the request.
403	Forbidden access	The accessKey does not match the profile of the user in the JWT.
404	Not Found	The resource or the endpoint does not exist.
409	Conflict	The user is trying to send a message to a person with an active out of office and outOfOfficeIgnored is set to false.
500	Internal server error	An unexpected error occurred, but the service is not down (to remove).
503	Service temporarily unavailable	The eHealthBox service is down.

7.3 Delivery failures

Mail delivery system error codes originating from the application:

These error codes indicate a problem with the message and or its recipients and are generated asynchronously. The sender of the message will receive this error message in his inbox.

Error code	Component	Description	Solution
700	Error	Unknown technical error.	Refer to the Contact Center
701	Error	Business validation error.	Check all fields, especially the recipients. The publication id can be found under *Publication id.*
702	Error	Duplicate publication id.	Chose a new publication id, and send again. The duplicate publication id can be found under *PublicationId*.
703	Error	One or more recipients are invalid.	Invalid recipients can be found under *Destination*.

Example:

```
{
   "content": {
       "size": 1849,
       "sender": {
           "actor": {
               "user": false,
               "organization": true,
               "organizationName": "Noreply"
           },
           "identifiers": {
               "entity": "12345678912",
               "quality": "CITIZEN",
               "entityType": "INSS"
           }
       },
       "annexes": [],
       "original": {
           "type": "ERROR",
           "title": "Delivery Status Notification (Failure)",
           "payload": "<!DOCTYPE html><meta http-equiv=\"Content-
Type\" content=\"text/html;charset=UTF-
8\"><html><body><h2>L'envoi du document suivant a échoué :<\/h2><br/><br/><b>T
itre du document : <\/b>K8WUO4J48MR59<br/><br/><b>Date de publication : <\/b>1
0:58:15, 18/10/2019<br/><br/><b>Détails techniques : <\/b>Les destinataires su
ivants ne sont pas reconnus par le système. Veuillez vérifier si l'ID du desti
nataire est correct et si le destinataire possède la qualité spécifiée.<br/><b
r/>L'ID est le numéro d'identification unique d'une personne ou d'une organisa
tion. Ce numéro peut être le numéro NISS (numéro de 11 chiffres qui se trouve
dans le coin supérieur droit de la carte SIS), le numéro NIHII (numéro INAMI d
e 8 chiffres) ou le numéro CBE (numéro d'entreprise de 10 chiffres).<br/><b>De
stinataire(s) : <\/b><br/>90000000000(INSS) - Dentiste<br/><b>Pièce(s) jointe(
s):<\/b><br/><br/>Titre : QW5uZXh0aXRsZQ==<br/>Nom du fichier : pdf.pdf<br/><b
r/><br/><br /><br /> -------------------------------
- <br /><br /><h2>Het verzenden van het volgende document is mislukt :<\/h2><b
r/><br/><b>Titel van het document : <\/b>K8WUO4J48MR59<br/><br/><b>Publicatied
atum : <\/b>10:58:15, 18/10/2019<br/><br/><b>Technische gegevens : <\/b>De vol
gende ontvangers worden niet herkend door het systeem. Controleer of de ID van
 de ontvanger correct is en of hij beschikt over de opgegeven bevoegdheid.<br/
><br/>De ID is het unieke identificatienummer van een persoon of van een organ
isatie. Dit nummer kan het INSZ-
nummer (nummer van 11 cijfers op de rechterbovenhoek van de SIS-
kaart), het NIHII-nummer (RIZIV-nummer van 8 cijfers) of het KBO-
nummer (ondernemingsnummer van 10 cijfers) zijn.<br/><b>Ontvanger(s) : <\/b><b
r/>90000000000(INSS) - Tandarts<br/><b>Bijlagen :<\/b><br/><br/>Titel : QW5uZX
h0aXRsZQ==<br/>Bestandsnaam  : pdf.pdf<br/><br/><br/><\/body><\/html>",
           "metadata": {
               "code": "703",
               "message": "One or more recipients are invalid.",
               "originalPublicationId": "97F2VD9C50RJ5"
           },
           "encrypted": false,
           "important": false,
           "extensions": {
               "applicationName": "eHboxSystem",
               "payloadFilename": "message.html",
               "undeliveredRecipients": [
                   {
                       "identifiers": {
                           "entity": "90000000000",
                           "quality": "DENTIST",
                           "entityType": "INSS"
                       },
                       "outOfOfficeIgnored": false
```

```
                    }
                ]
            },
            "recipients": [
                {
                    "person": {
                        "ssin": "90000000000",
                        "lastName": "Everybody",
                        "firstName": "Jason"
                    },
                    "identifiers": {
                        "entity": "90000000000",
                        "quality": "DOCTOR",
                        "entityType": "INSS"
                    },
                    "outOfOfficeIgnored": false
                }
            ],
            "publicationId": "1571389102801",
            "annexesMetadata": [],
            "payloadMimetype": "text/html",
            "acknowledgements": {
                "read": false,
                "sent": false,
                "viewed": false
            }
        },
        "recipient": {
            "person": {
                "ssin": "90000000000",
                "lastName": "Everybody",
                "firstName": "Jason"
            },
            "identifiers": {
                "entity": "90000000000",
                "quality": "DOCTOR",
                "entityType": "INSS"
            },
            "outOfOfficeIgnored": false
        },
        "identifier": 3000002876553,
         "expirationDate": "2022-12-14",
         "expirationBinDate": "2022-07-15",
         "expirationSentDate": "2022-07-15",
         "publicationDateTime": "2022-06-15T18:47:48.849655",
         "expirationBinsentDate": "2022-07-15",
         "expirationStandbyDate": "2022-12-14",
    },
    "metadata": {
        "viewDateTime": "2019-10-18T11:49:25.520882"
    }
}
```

7.4 Business error codes

Mail delivery system error codes originating from the application.

These error codes indicate a problem with the message and/or its recipients and are generated synchronously. The API can send the following errors as response:

Code	Message	Description
INVALID_FOLDER	Folder xxxx is wrong. Must be a value in [in, bin, binsent, sent]	The folder introduced is not correct. It has to be in, bin, binsent or sent
ANNEX_NOT_FOUND	Attachment with accessKey {}, folder {}, messageId {}, key {} was not found.	The introduced attachment key cannot be found in the specified message, box and folder or the attachmentKey is wrong.
400 BAD_REQUEST	Malformed Json request	The message structure of the request is not correct. Please check the detailed message for more information.
801	The message exceeds the maximum authorized size.	Please lower the message size by deleting some appendixes or by splitting the message and send the message again. Please take into account that encryption may have increased the total message size.
803	INVALID_ARGUMENT: Invalid identifier Quality with value { }	The specified Quality is invalid; please verify that Quality is a quality recognized by the system.
806	Message with id {} not exist in the mailbox {} in folder {}	The specified message is not present in the specified folder.
810	"INVALID_ARGUMENT: Invalid identifier: should (only) contain 'entity', 'entityType' and 'quality'.	The identifier object of an recipient does not contain the 3 allowed identifiers (entity, entityType and quality) or contains more identifiers.
814	Requested boxId is not owned by user	The accessKey does not match the profile of the user in the JWT.
814	Connected user is not the owner of this nihii	The user is trying to send a message with a NIHII that is not owned by the user.
816	"hash mismatch. Expected : {}, actual: {}",	The digest of the streamed document does not match with the supplied one.
820	the period xx/xx/xxxx to xx/xx/xxxx is invalid because it overlaps another period.	The out of office periods cannot overlap each other's.
821	The end of the period cannot be further than a year in the future.	The end of the period cannot be further than a year in the future.

822	The start date can't be after the end date.	The start date cannot be after the end date.
823	The start date can't be in the past.	The start date cannot be in the past.
824	One or more substitutes cannot be chosen because they are absent.	One of the substitutes has an active out of office.
825	The number of substitutes may not exceed 5.	The number of substitutes may not exceed 5 substitutes.
826	One or more recipients have an Out-Of-Office active. For this error, there is also an additional array: **recipientsInError** The number of out of office for one eHealthBox may not exceed 10.	One or more recipients have an Out-Of-Office active.
827	One or more substitutes are unknown or wrong, please correct them.	One or more substitutes are unknown or wrong.
828	The user is unknown or not correct, please correct him.	The user of the box is unknown or wrong.
829	A valid substitute is a person, not an organization.	An organization cannot be a substitute.
840	The OutOfOffice xxxxxxx does not exist for the accesskey yyyyyyyy	The out of office does not exist.
830	A person cannot be substitute for himself.	The user has put himself as substitute.
900	The document type is incorrect.	Type is NEWS, ACKNOWLEDGMENT or ERROR
901	One of the encryptable fields is not in base64 (with padding) format.	One of the following fields is not Binary 64 when the flag is encrypted true: payload freeText leftCell rightCell oldFreeInformation.value patientNiss annexesMetadate.title
902	The payload mimetype must match text or html mimetype.	mimetype is not "text/html"
904	Metadata's key or value cannot be empty?	Metadata key or value is empty.
905	Health meta must contains no-blank values.	eHealthMeta key or value is empty.
906	INVALID_ARGUMENT: The applicationName should be between 1 and 25 characters.	The applicationName is empty or longer than 25 characters.
907	The message exceed the limit of total annexes count.	Reduce the number to maximum 25.
MISSING_ATTACHMENT	Misses match(es) between message and attachments for files:	An attachment referred to in annexes does not have a matching file.

MISSING_ATTACHMENT_METADATA	Misses match(es) between message, annexes, Metadata and attachments for files:	An attachment referred in annexes and metadata does not have a matching file.
CONTENT_NOT_ENCODED	Field xxxx is not encoded in base64 (with padding).	The mentioned field is not encoded in base 64 with padding. This is mandatory if it is an encrypted message (encrypted tag true).
DUPLICATE_ATTACHMENT	Request contains duplicate attachment part names	The contentId of one of the annexes is not unique in the request .

Example:

```
{
   "title": "Not found",
   "detail": "Folder xxxx isn't correct. Must be a value in [in, bin,
binsent, sent]",
   "instance": "845ccc229ed0a4f5",
   "code": "INVALID_FOLDER"
}
```

8. Annexes

8.1 Java snippet send message with annex

```
package be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices;

import static java.nio.charset.StandardCharsets.UTF_8;
import static java.util.Collections.singletonList;
import static org.springframework.http.HttpMethod.POST;
import static org.springframework.http.MediaType.MULTIPART_FORM_DATA;

import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.EhboxException;
import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.SimpleEhboxMessage;
import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.SimpleEhboxMessage.Ac
knowledgements;
import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.SimpleEhboxMessage.An
nexMetadata;
import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.SimpleEhboxMessage.Id
entifiers;
import be.smals.vas.hfcse.pcrtestprescription.infrastructure.externalservices.dto.SimpleEhboxMessage.Re
cipient;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.net.URI;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Collections;
import java.util.Map;
import java.util.Set;
import java.util.UUID;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBuilder;

@Service
public class EHBoxServiceImpl {

  private static final String INSTITUTION_ENTITY_TYPE = "CBE";
  private static final String INSTITUTION_QUALITY_NAME = "INSTITUTION";
  private static final String ATTACHMENT_FILE_NAME = "Attachment_{timestamp}.xml";
  private static final String MESSAGE_TYPE = "DOCUMENT";
  private static final String MULTIPART_BODY_PART_NAME = "body";
  private static final String BODY_FILE_NAME = "body.json";

  private static final String MESSAGE_TITLE = "Message title";
  private static final String MESSAGE_PAYLOAD = "<p>You received a new message (see annexe).</p>";
  private static final String ATTACHMENT_TITLE = "Attachment";
  private static final String PUBLICATION_PATH = "/<access_key>/publications";
  private static final String META_DATA_CM_ATTACHMENT_TRANSPORT_TYPE = "CM-AttachmentTransportType";
  private static final String META_DATA_EFORMS = "EFORMS";
  private static final String META_DATA_HC_FUNCTIONAL_TYPE = "HC-FunctionalType";
  private static final String META_DATA_HC_ATTACHMENT_FILENAME = "HC-AttachmentFilename";

  private final RestTemplate restTemplate;
  private final ObjectMapper mapper;
  private final String eHBoxApiScheme;
  private final String eHBoxApiHost;
  private final String eHBoxApiPort;
  private final String eHBoxApiPath;

  public EHBoxServiceImpl(final RestTemplate restTemplate,
                          final ObjectMapper mapper,
                          @Value("${vas.ehbox-api.scheme}") final String eHBoxApiScheme,
                          @Value("${vas.ehbox-api.host}") final String eHBoxApiHost,
                          @Value("${vas.ehbox-api.port}") final String eHBoxApiPort,
                          @Value("${vas.ehbox-api.path}") final String eHBoxApiPath) {
    this.restTemplate = restTemplate;
    this.mapper = mapper;
    this.eHBoxApiScheme = eHBoxApiScheme;
    this.eHBoxApiHost = eHBoxApiHost;
    this.eHBoxApiPort = eHBoxApiPort;
    this.eHBoxApiPath = eHBoxApiPath;
  }

  public void sendMessageToCbe(final String cbeNumber, final String attachment) {

    final URI url = UriComponentsBuilder.newInstance()
      .scheme(eHBoxApiScheme)
      .host(eHBoxApiHost)
      .port(eHBoxApiPort)
      .path(eHBoxApiPath + PUBLICATION_PATH)
      .build().toUri();

    final var requestEntity = buildHttpEntity(cbeNumber, attachment);
```

```java
    final ResponseEntity<Void> responseEntity = restTemplate.exchange(url, POST, requestEntity, Void.class);

    if (!responseEntity.getStatusCode().is2xxSuccessful()) {
      throw new EhboxException("Error when calling Ehbox integration service: status = " + responseEntity.getStatusCodeValue());
    }

  }

  private HttpEntity<MultiValueMap<String, Object>> buildHttpEntity(final String cbeNumber,
                                                                    final String attachment) {
    final var body = new LinkedMultiValueMap<String, Object>();

    final var timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMddHHmmssS"));
    final var attachmentFilename = ATTACHMENT_FILE_NAME.replace("{timestamp}", timestamp);
    final var attachmentContentId = UUID.randomUUID().toString();

    try {

      final var simpleEhboxMessage = buildEhboxMessage(cbeNumber, attachmentFilename, attachmentContentId);
      body.add(MULTIPART_BODY_PART_NAME, getResourceHttpEntity(mapper.writeValueAsString(simpleEhboxMessage).getBytes(UTF_8),
                                                               BODY_FILE_NAME,
                                                               MediaType.APPLICATION_JSON));
    } catch (JsonProcessingException e) {
      throw new EhboxException("Error while serializing EhboxMessage", e);
    }

    body.add(attachmentContentId, getResourceHttpEntity(attachment.getBytes(UTF_8), attachmentFilename, MediaType.APPLICATION_XML));
    return new HttpEntity<>(body, createHeaders());
  }

  private SimpleEhboxMessage buildEhboxMessage(final String cbeNumber,
                                               final String attachmentFilename,
                                               final String attachmentContentId) {

    final Identifiers recipientIdentifiers = resolveRecipientIdentifiers(cbeNumber);

    final var xmlAnnexMetadata = getAnnexMetadata(attachmentContentId,
                                                  attachmentFilename,
                                                  MediaType.APPLICATION_XML_VALUE,
                                                  ATTACHMENT_TITLE);

    return SimpleEhboxMessage.builder()
```

```java
      .type(MESSAGE_TYPE)
      .title(MESSAGE_TITLE)
      .recipients(Collections.singleton(Recipient.builder()
                                         .identifiers(recipientIdentifiers)
                                         .build()))
      .payload(MESSAGE_PAYLOAD)
      .payloadMimetype(MediaType.TEXT_HTML_VALUE)
      .annexesMetadata(Set.of(xmlAnnexMetadata))
      .acknowledgements(Acknowledgements.builder().read(false).sent(false).viewed(false).build())
      .encrypted(false)
      .important(false)
      .metadata(getMetadata(attachmentFilename))
      .publicationId(String.valueOf(System.currentTimeMillis()))
      .build();

  }

  private Map<String, String> getMetadata(final String attachmentFilename) {
    return Map.of(
      META_DATA_CM_ATTACHMENT_TRANSPORT_TYPE, META_DATA_EFORMS,
      META_DATA_HC_FUNCTIONAL_TYPE, META_DATA_EFORMS,
      META_DATA_HC_ATTACHMENT_FILENAME, attachmentFilename
    );
  }

  private Identifiers resolveRecipientIdentifiers(final String cbeNumber) {
    return Identifiers.builder()
      .entity(cbeNumber)
      .entityType(INSTITUTION_ENTITY_TYPE)
      .quality(INSTITUTION_QUALITY_NAME)
      .build();
  }

  private AnnexMetadata getAnnexMetadata(final String contentId,
                                         final String attachmentFilename,
                                         final String contentType,
                                         final String title) {
    return AnnexMetadata.builder()
      .contentId(contentId)
      .contentType(contentType)
      .fileName(attachmentFilename)
      .digest("") // TODO compute digest
      .title(title)
      .build();
  }
  private HttpEntity<Resource> getResourceHttpEntity(final byte[] content, final String fileName, final
 MediaType contentType) {
    final var file = new ByteArrayResource(content) {
      @Override
      public String getFilename() {
        return fileName;
      }
    };
    final var xmlHeaders = new HttpHeaders();
    xmlHeaders.setContentType(contentType);
    return new HttpEntity<>(file, xmlHeaders);
  }

  private HttpHeaders createHeaders() {
    return new HttpHeaders() {{
      setContentType(MULTIPART_FORM_DATA);
      setAccept(singletonList(MediaType.ALL));
    }};
  }

}
```

8.2 cURL example request

Below an example of a request, in a cURL format (HIDDEN_DATA should be replaced with the relevant id and bearer token). That examples assumes the integrator already has the necessary rights to query the API, send messages, obtain Bearer tokens…

Example :

```
curl 'https://api-intr2.ehealth.fgov.be/ehBox/mailboxes/HIDDEN_DATA/publications' \
  -H 'Accept: application/json, text/plain, */*' \
  -H 'Accept-Language: en-US,en;q=0.9' \
  -H 'Authorization: Bearer HIDDEN_DATA' \
  -H 'Connection: keep-alive' \
  -H 'Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKXBXg0SZCzeUZVRG' \
  --data-raw $'------WebKitFormBoundaryKXBXg0SZCzeUZVRG\r\nContent-Disposition: form-data;
name="body"; filename="blob"\r\nContent-Type:
application/json\r\n\r\n{"acknowledgements":{"read":false,"sent":false,"viewed":false},"encryp
ted":false,"important":false,"metadata":{},"payload":"<p>Message
Content</p>\\n","payloadMimetype":"text/html","publicationId":"84bntmibap8go","recipients":[{"identifiers":{"entityType":"INSS","e
ntity":"92103029927","quality":"DOCTOR"},"outOfOfficeIgnored":false}],"title":"Message
Title","type":"DOCUMENT","extensions":{"applicationName":"WEBAPP"},"annexesMetadata":[{"conten
tId":"file-6432685368","fileName":"attachment.txt","title":"attachment.txt"}]}\r\n------
WebKitFormBoundaryKXBXg0SZCzeUZVRG\r\nContent-Disposition: form-data; name="file-6432685368";
filename="attachment.txt"\r\nContent-Type: text/plain\r\n\r\n\r\n------
WebKitFormBoundaryKXBXg0SZCzeUZVRG--\r\n' \
  --compressed
```

Please note that If the message contains an attachment, the expected request takes the form of a POST with content-type multipart/form-data

- The multipart contains (2*N)+1 items, N being the amount of attachments sent, and the “+1” is the JSON content of the request (the message, with recipients, etc).

- Each attachment is present in a form-data part with a random name, preferably generated (avoid using the real file name, as the user could upload multiple files with the same name), each attachment's name If referred to in the "body".
- Each attachment also has a form-data part that informs of its filename.
- The actual content of the request should be in a form-data part JSON, with Content-Type: application/json.