|  |
| --- |
| Federated eBox for citizens  API RESTful definition : Message Registry Contracts (interfaces) |

SpecifIcation DOCUMENT 1.0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Statut** | **Date** | **Auteur(s)** | **Nature des modifications** |
| 0.1 | Draft | 2017-06-20 | Antoine Moulart  Kevin Noppe | First version |
| 0.2 | Draft v0.2 | 2017-07-05 | Antoine Moulart  Kevin Noppe | Add E-R schema ; add ‘attachment’ resource;  Update & detail operations descriptions & examples (+update JSON-schemas: *appendix)* |
| 0.3 | Proposal | 2017-07-19 | Antoine Moulart  Kevin Noppe | Review architect (D. Morer) ; distinction ‘list of resourceSummary’ and ‘specific resource’ + comments BOSA (meeting 18/07) ; separate /sender into /senderOrganization and /senderApplication |
| 0.4 | Proposal | 2017-08-03 | Antoine Moulart  Kevin Noppe | Minor updates after internal review + add schema (overview of access to resources). |
| 0.5 | Proposal | 2017-08-08 | Antoine Moulart  Kevin Noppe | Add list of specific error codes  + change logos & attachments handling |
| 1.0 | First version | 2017-12-20 | Antoine Moulart  Kevin Noppe | Finalization of version 1.0 of the API to be used for the MVP |
| 1.1 | Minor update | 2018-05/24 | Antoine Moulart  Kevin Noppe | Add registeredMail and expirationDate to messageSummary + update messages search criteria |

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewers** | **Nom** | **Version reviewée** | **Commentaires** |
| SPOC Client  (Responsable Business) | Claudia Laeremans (BCSS-KSZ)  Eric Descamps (BOSA) |  |  |
| Gestionnaire Clients | Donald De Keyser |  |  |
| Chef de Projet | Jérôme Vos |  |  |
| CPL | Roel Biermans |  |  |
| Chain Service Manager | Catherine Schoetter |  |  |
| Domain Architect for BOSA DT - PGA | Arnaud Reper | 0.1, 0.2, 0.3 |  |
| REST-team Smals (Enterprise & Solutions Architects) | A. Semal – S. Flamme – W. Salembier | 0.3 |  |
|  |  |  |  |

Contents

[Introduction 3](#_Toc499028773)

[General principles 3](#_Toc499028774)

[Resources and Operations 4](#_Toc499028775)

[Resources description 4](#_Toc499028776)

[JSON schemas 5](#_Toc499028777)

[Entity Relationship Diagram 5](#_Toc499028778)

[Operations 6](#_Toc499028779)

[GET **/api** 6](#_Toc499028780)

[GET **/ebox** 7](#_Toc499028781)

[GET ebox**/messages**?{*param*} 8](#_Toc499028782)

[GET ebox/messages**/{$messageId}** 11](#_Toc499028783)

[GET ebox/messages/{$messageId}**/attachments** 12](#_Toc499028784)

[GET ebox/messages/{$messageId}/attachments**/{$attachmentId}** 13](#_Toc499028785)

[GET ebox/messages/{$messageId}/attachments/{*$attachmentId*}**/content** 14](#_Toc499028786)

[Overview of access to resources 15](#_Toc499028787)

[GET referenceData**/messageTypes**?{*param*} 16](#_Toc499028788)

[GET referenceData/messageTypes**/{$messageTypeId}** 18](#_Toc499028789)

[GET referenceData**/senderOrganizations**?{*param*} 19](#_Toc499028790)

[GET referenceData/senderOrganizations**/{$organizationId}** 21](#_Toc499028791)

[GET referenceData**/senderApplications**?{*param*} 23](#_Toc499028792)

[GET referenceData/senderApplications**/{$senderApplicationId}** 25](#_Toc499028793)

[Out-of-scope (MVP) 26](#_Toc499028794)

[Standard REST Conventions and common elements 27](#_Toc499028795)

[Resource URI 27](#_Toc499028796)

[Reserved words - Query parameters 27](#_Toc499028797)

[JSON properties 28](#_Toc499028798)

[Link references 28](#_Toc499028799)

[Collection 29](#_Toc499028800)

[List of resources links 29](#_Toc499028801)

[List of resources 30](#_Toc499028802)

[Filtered collection 31](#_Toc499028803)

[Paging over a large collection 32](#_Toc499028804)

[Common Status codes 34](#_Toc499028805)

[1xx Informational 34](#_Toc499028806)

[2xx Success 34](#_Toc499028807)

[3xx Redirection 35](#_Toc499028808)

[4xx Client Error 36](#_Toc499028809)

[5xx Server Error 38](#_Toc499028810)

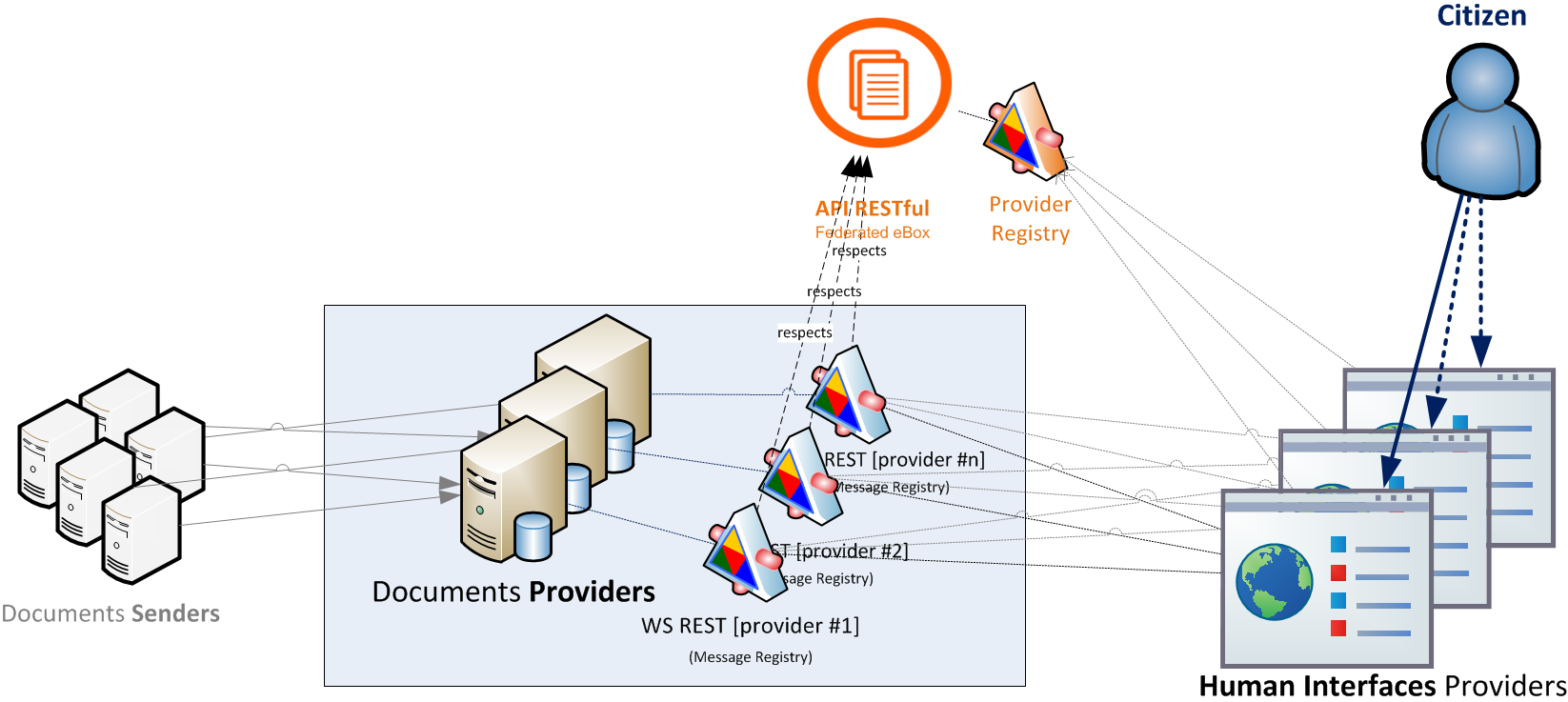
[Specific Error Codes 39](#_Toc499028811)

[Error handling : complete examples 40](#_Toc499028812)

# Introduction

This document aims to describe a first proposal of the **RESTful API** designed as the **Message Registry Contracts** of the **Federated eBox for citizens**. The goal of this API is to standardize and facilitate ebox integration between the different partners:

* + **Documents Providers** will have to respect this API to develop their Message Registry (WS REST);
  + **Human Interfaces Providers** will integrate the different *REST services* in the same way(using the *Provider Registry* to find available WS).



The first section describes the different resources. The second section gives all the operations details, with examples for each operation. A specific section summarizes the elements considered as out-of-scope for the first API version (MVP).

In the next chapters, we summarize the main standardized conventions and common elements used to design the RESTful API, as well as the list of the common status codes.

## General principles

* The identification of the given citizen (ebox) is deducted from the OAuth2 AccessToken (via introspect call on the Authorization Server) used to call the REST service (aka *MessageRegistry)*.
* Minimum Viable Product (MVP) supports GET operations only
  + GET /resources **🡪 full list of ‘resourceSummary’** (= resource with main attributes)
  + GET /resources?{param} **🡪 filtered list of ‘resourceSummary’** (= resource with main attributes)
  + GET /resources/resourceId **🡪 specific resource** (= resource with all details)
* Reserved words parameters, such as those used to manipulate collection-type resources, or paging mechanisms, must comply with the conventions detailed in the sections below.
* The use of the status codes must also respect the HTTP standard provided by the IETF (cf. specific section below).

# Resources and Operations

First of all, we begin with a simple description of each identified resource.

## Resources description

|  |  |
| --- | --- |
| Resource | Description |
| api | API is a standard resource used to provide links to the different business resources, and also to the Swagger Specification file. |
| ebox | An ebox represents the electronic post-box, made available by a given MessageRegistry Provider, for a given citizen. |
| messages | Messages are the main elements contained in an ebox. A message is basically composed of a set of metadata (e.g. subject, receipt date, etc.), one body, one or more attachments. |
| attachments | An attachment represents the content part of a message when available as a downloadable file, such as a PDF.  A message can also contain a body which can also act as the content part of a message. Furthermore, a message can contain both an array of attachments as well as a body. Both elements are available in the content element of a message. |
| messageTypes | MessageTypes are the different types of messages defined by the senders. A messageType usually corresponds to a business category of a letter or document (e.g. attestation *X.*, declaration *Y.*, communication *Z., etc.)*.  NB: a message must always be linked to one (and only one) messageType. |
| senderOrganizations | A senderOrganization is the organization (Institution or Company) at the origin of a message[[1]](#footnote-2). Each senderOrganization is identified by its Company Number (num. BCE / KBO num.).  NB: a message must always be linked to one (and only one) senderOrganization. |
| senderApplications | A senderApplication is the business application that could be at the origin of a message published in ebox. A senderApplication usually corresponds to an online service available for the citizen.  NB: a message could be linked to one (and only one) senderApplication. It is also possible that a message is not linked to any senderApplication. |

### JSON schemas

JSON files in appendix.

### Entity Relationship Diagram



Entity relationships:

* An **ebox** is linked to 1..1 **citizenProfile** [*out-of-scope in this specification*]
* An **ebox** contains 0..n **messages**
* A **message** contains 0..n **attachment(s)**
* A **message** is linked to 1..1 **messageType**
* A **message** is linked to 1..1 **senderOrganization**
* A **message** is linked to 0..1 **senderApplication**
* A **messageType** is linked to 0..n **senderOrganization** ; a **senderOrganization** is linked to 0..n **messageType**;
* A **messageType** is linked to 0..n **senderApplication** ; a **senderApplication** is linked to 0..n **messageType;**
* A **senderApplication** is linked to 0..n **senderOrganization** ; a **senderOrganization** is linked to 0..n **senderApplication**

## Operations

This section describes the different operations on each identified resource.

### GET **/api**

Retrieve documentation and entry points for this API.

|  |  |  |
| --- | --- | --- |
| Response | | |
| body | The api response contains links to the API entry points.   * Link to the root collections * Link to the HTML documentation * Link to the Swagger specification | ​​​{  "\_links": {  "resource:ebox": {  "href": "/REST/ebox/citizen/v1/ebox"  },  "resource:messages": {  "href": "/REST/ebox/citizen/v1/messages"  },  "resource:messageTypes": {  "href": "/REST/ebox/citizen/v1/messageTypes"  },  "resource:senders": {  "href": "/REST/ebox/citizen/v1/senders"  },  "doc": {  "href": "/REST/ebox/citizen/v1/api/doc/html",  "type": "text/html"  },  "swagger": {  "href": "/REST/ebox/citizen/v1/api/doc/swagger",  "type": "application/json"  }  }  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/#http-200) | OK | Default success code ​ |

### GET **/ebox**

Get the main information of an ebox for a given citizen.

|  |  |  |
| --- | --- | --- |
| Response | | |
| body | The ebox response contains:   * numberOfMessages [**required**] [integer] * numberOfUnreadMessages [**required**] [integer] * lastReceiptDate [**optional**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * lastConsultationDate [**optional**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * eboxSize [**optional**] [integer] (size in kB) | {  "numberOfMessages": 3,  "numberOfUnreadMessages": 1,  "lastConsultationDate": "2017-07-03T09:26:00Z",  "lastReceiptDate": "2017-06-12T10:15:00Z",  "eboxSize": 3200  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code. |

**NB:** if the ebox for the given citizen (authenticated via the OAuth2 AccessToken) is not yet activated at the MessageRegistry Provider, the expected response should be 200. **Please, do not return an error code in this case.** The body response will simply indicate ‘numberOfMessages = 0’.

### GET ebox**/messages**?{*param*}

Get the list of messages available in a given ebox, possibly filtered with different criteria.

In the response, each message is provided with its main attributes (=messageSummary). To get the full details of a given message, see next operation with use of the messageId in the path-param.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| receivedBefore  [full-date format: yyyy-mm-dd] | query-param | Select only the messages received before the given date.  ?receivedBefore=2017-07-01 |
| receivedAfter  [full-date format: yyyy-mm-dd] | query-param | Select only the messages received after the given date.  ?receivedAfter=2017-07-01 |
| expiredBefore  [full-date format: yyyy-mm-dd] | query-param | Select only the messages that will expire before the given date. ?expiredBefore=2017-07-01 |
| expiredAfter  [full-date format: yyyy-mm-dd] | query-param | Select only the messages that will expire after the given date. ?expiredAfter=2020-01-01 |
| readStatus  [boolean] | query-param | Boolean used to indicate that the message has been read.  ?readStatus=false |
| registeredMail  [boolean] | query-param | Boolean used to indicate whether or not a message is a registered mail. ?registeredMail=true |
| messageTypeId [string] | query-param | Select only the messages linked to the given messageTypeId. ?messageTypeId=PatientProof |
| senderOrganizationId [string] | query-param | Select only the messages linked to the given senderOrganizationId. ?senderOrganizationId=0928100313 |
| senderApplicationId [string] | query-param | Select only the messages linked to the given senderApplicationId. ?senderApplicationId=myminfin |
| sort  [string]  *[default: sort=‑receiptDate]* | query-param | Comma-separated list of properties to sort data. The default sorting direction is ascending. Names may be prefixed with ‘-‘ to sort descending.  ?sort=receiptDate |
| page [number]  *[default: page=1]* | query-param | Use this parameter to request a specific page. Page numbers are 1-based.  ?page=3 |
| pageSize [number]  *[default: pageSize=25]* | query-param | Use this parameter to specify the page size.  ?pageSize=12 |
| subject [string] | query-param | Filter only titles (subject) that contain a specific word. |
| Response | | |
| body | The response is a **collection** JSON object. Each item corresponds to the resource type “messageSummary”.  A **messageSummary** is composed of the following attributes:   * messageId [**required**] [string] * subject[**required**] [translatedString] * receiptDate [**optional**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * expirationDate [**required**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * readStatus [**required**] [boolean] * registeredMail [**required**] [boolean] * messageTypeId [**required**] [string] * senderOrganizationId [**required**] [string] * senderApplicationId [**optional**] [string]   **sort by default of collection:**  **–** *receiptDate (descending)* | {  "items": [{  "messageId": "9D000003NG00U",  "subject": { "nl": "Aanvraag pensioengegevens",  "fr": "Demande données de pension",  "de": "Anfrage Rentendaten",  "en": "Request pension data"  },  "receiptDate": "2017-05-20T10:15:00Z",  "readStatus": false,  "messageTypeId": "PensionAttest",  "senderOrganizationId": "0123456789",  "href": "./messages/9D000003NG00U"  },  {  "messageId": "16cd9678-5687-11e7-907b-a6006ad3dba0",  "subject": {  "nl": "Pensioen attest",  "fr": "Attestation de pensions"  },  "receiptDate": "2017-05-24T09:26:00Z",  "readStatus": true,  "messageTypeId": "PensionAttest",  "senderOrganizationId": "0123456789",  "senderApplicationId": "mypension:citizen",  "href": "./messages/16cd9678-5687-11e7-907b-a6006ad3dba0"  },  {  "messageId": "16cd9678-5687-11e7-907b-a6006ad3dba0",  "subject": {  "nl": "Verpleegkundige afrekening - Q2 2017",  "fr": "Compte infirmier - Q2 2017",  "de": "Compte infirmier - Q2 2017",  "en": "Infirmary bill - Q2 2017"  },  "receiptDate": "2017-06-12T10:15:00Z",  "readStatus": false,  "messageTypeId": "PatientProof",  "senderOrganizationId": "0206731645",  "senderApplicationId": "infirmarySystem",  "href": "./messages/16cd9678-5687-11e7-907b-a6006ad3dba0"  }],  "totalItems": 3  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### GET ebox/messages**/{$messageId}**

Get the full details of a given message.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| messageId [string] | path-param | Technical ID (e.g. UUID) uniquely identifying the message. |
| Response | | |
| body | A **message** is composed of the following attributes:   * messageId [**required**] [string] * subject[**required**] [translatedString] * receiptDate [**required**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * expirationDate [**optional**] [date-time format: yyyy-MM-dd’T’HH:mm:ssXXX] * readStatus [**required**] [boolean] * registeredMail [**required**] [boolean] * messageTypeId [**required**] [string] * senderOrganizationId [**required**] [string] * senderApplicationId [**optional**] [string] * content [**anyOf**] [**required**]   + body [**optional**] [translatedString]   + attachments [0..\*] [**optional**] [list] * paymentData **[optional**] [object] * labels **[optional**] [list] | {  "messageId": "16cd9678-5687-11e7-907b-a6006ad3dba0",  "subject": {  "nl": "Pensioen attest",  "fr": "Attestation de pensions"  },  "receiptDate": "2017-05-24T09:26:00Z",  "expirationDate": "2019-05-24T09:26:00Z",  "readStatus": true,  "registeredMail": false,  "messageTypeId": "PensionAttest",  "senderOrganizationId": "0123456789",  "senderApplicationId": "mypension:citizen",  "content": {  "attachments": {  "items": [{  "attachmentId": "599afe06-d3f9-4a67-ac84-97a9172ea8b5",  "attachmentTitle": {  "nl": "Bestand pensioenkadaster.pdf",  "fr": "Fichier cadastre pension.pdf"  },  "mediaType": "application/pdf",  "size": 178,  "digest": {  "digestValue": "szOe0ZbpGXDLMT9vs1dsoJLM0hgkVCcrQyAJpfFt6FtwQTl5wPdTVZfdoKh73/u7ZFiFgUHE15LqMvhV6BLCLg==",  "digestMethod": "SHA-512"  },  "attachmentSigned": true,  "href": "./16cd9678-5687-11e7-907b-a6006ad3dba0/attachments/599afe06-d3f9-4a67-ac84-97a9172ea8b5"  }],  "totalItems": 1  }} |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code. |

### GET ebox/messages/{$messageId}**/attachments**

Get the metadata of the attachments linked to a given messages

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| messageId [string] | path-param | Technical ID (e.g. UUID) uniquely identifying the message. |
| Response | | |
| body | The response is a **collection** JSON object. Each item corresponds to the resource type “attachment”.  An **attachment** is composed of the following attributes:   * attachmentId [**required**] [string] * attachmentTitle **[required**] [translatedString] * mediaType [**optional**] [string] * size [**required**] [integer] * digest [**optional**] [string] * attachmentSigned [**optional**] [boolean] | ​​​{  "items": [{  "attachmentId": "a6006ad3dba0",  "attachmentTitle": {  "nl": "attest.pdf",  "fr": "attestation.pdf",  "de": "attest.pdf",  "en": "attest.pdf"  },  "attachmentSigned": false,  "mediaType": "application/pdf",  "digest": {  "digestValue": "szOe0ZbpGXDLMT9vs1dsoJLM0hgkVCcrQyAJpfFt6FtwQTl5wPdTVZfdoKh73/u7ZFiFgUHE15LqMvhV6BLCLg==",  "digestMethod": "SHA-512"  },  "size": 25600  }],  "totalItems": 1  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code. |

### GET ebox/messages/{$messageId}/attachments**/{$attachmentId}**

Get the metadata of a given attachment, for a given message.

NB: to get the content self of the attachment, see next operation.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| messageId [string] | path-param | Technical ID (e.g. UUID) uniquely identifying the message. |
| attachmentId [string] | path-param | Id of the attachment |
| Response | | |
| body | The **attachment** response contains:   * attachmentId [**required**] [string] * attachmentTitle **[required**] [translatedString] * mediaType [**optional**] [string] * size [**required**] [integer] * digest [**optional**] [string] * attachmentSigned [**optional**] [boolean] | ​​​{  "type": "attachment",  "attachmentId": "a6006ad3dba0",  "attachmentTitle": {  "nl": "attest.pdf",  "fr": "attestation.pdf",  "de": "attest.pdf",  "en": "attest.pdf"  },  "attachmentSigned": false,  "mediaType": "application/pdf",  "digest": {  "digestValue": "szOe0ZbpGXDLMT9vs1dsoJLM0hgkVCcrQyAJpfFt6FtwQTl5wPdTVZfdoKh73/u7ZFiFgUHE15LqMvhV6BLCLg==",  "digestMethod": "SHA-512"  },  "size": 25600  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code. |

### GET ebox/messages/{$messageId}/attachments/{*$attachmentId*}**/content**

Get an attachment (file content) for a given message.

This file content will be returned as an **application/octet-stream**.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| messageId [string] | path-param | Technical ID (e.g. UUID) uniquely identifying the message. |
| attachmentId [string] | path-param | Id of the attachment |
| body | The attachment content, as a binary file **(application/octet-stream).** | *$bytes* |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code. |

### Overview of access to resources



### GET referenceData**/messageTypes**?{*param*}

Get the list of messageTypes, possibly filtered with different criteria. Each element of the list is a *messageTypeSummary* that gives the main attributes about a messageType. In order to get the full details about a given messageType, we can use the next operation by providing the *messageId* as path param.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| senderOrganizationId  [string] | query-param | Filter only messageTypes that can be published by the senderOrganization represented by the given organizationId. |
| senderApplicationId  [string] | query-param | Filter only messageTypes that can be published by the senderApplication represented by the given applicationId. |
| name  [string] | query-param | Filter only messageTypes that contain a specific string in the name, description or identifier. |
| sort  [string]  *[default: sort=messageTypeId]* | query-param | Comma-separated list of properties to sort data. The default sorting direction is ascending. Names may be prefixed with ‘-‘ to sort descending.  ?sort=-messageTypeNameFr |
| page [number]  *[default: page=1]* | query-param | Use this parameter to request a specific page. Page numbers are 1-based.  ?page=3 |
| pageSize [number]  *[default: pageSize=25]* | query-param | Use this parameter to specify the page size.  ?pageSize=12 |
| Response | | |
| body | The response is a **collection** JSON object. Each item corresponds to the resource type “messageType”.  A **messageTypeSummary** is composed of the following attributes:   * messageTypeId [**required**] [string] * messageTypeName [**required**] [translatedString] * senderOrganizationIds [0..\*] [**optional**] [list] * senderApplicationIds [0..\*] [**optional**] [list]   **sort by default of collection:**  **+** *messageTypeId* *(ascending)* | {  "items": [  {  "messageTypeId": "PensionAttest",  "messageTypeName": {  "nl": "MessageType Pensioen attest",  "fr": "MessageType Attestation de pensions",  "de": "MessageType Attestation de pensions",  "en": "MessageType Pension file"  },  "senderOrganizationIds": [  "0123456789"  ],  "href": "./messageTypes/PensionAttest"  },  {  "messageTypeId": "PatientProof",  "messageTypeName": {  "nl": "Bewijsstuk patiënt",  "fr": "Justificatif patient",  "de": "Beweis Patient",  "en": "Patient proof"  },  "senderOrganizationIds": [  "0206731645"  ],  "senderApplicationIds": [  "infirmarySystem"  ],  "href": "./messageTypes/PatientProof"  }  ],  "totalItems": 2  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### GET referenceData/messageTypes**/{$messageTypeId}**

Get the details about a given messageType, identified by its unique *messageTypeId*.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| messageTypeId  [string] | path-param | ID of the messageType. |
| Response | | |
| body | A **messageType** is composed of the following attributes:   * messageTypeId [**required**] [string] * messageTypeName [**required**] [translatedString] * messageTypeDescription [**optional**] [translatedString] * validityPeriod [**optional**] [number] * businessMetadata [**optional**] [list] * senderOrganizationIds [0..\*] [**optional**] [list] * senderApplicationIds [0..\*] [**optional**] [list] | {  "messageTypeId": "PatientProof",  "messageTypeName": {  "nl": "Bewijsstuk patiënt",  "fr": "Justificatif patient",  "de": "Beweis Patient",  "en": "Patient proof"  },  "messageTypeDescription": {  "nl": "MessageType Description Bewijsstuk patiënt",  "fr": "MessageType Description Justificatif patient",  "de": "MessageType Description Beweis Patient",  "en": "MessageType Description Patient proof"  },  "validityPeriod": {  "validityPeriodNumber": 1,  "validityPeriodUnit": "year"  },  "senderOrganizationIds": [  "0206731645"  ],  "senderApplicationIds": [  "infirmarySystem"  ]  } |

### GET referenceData**/senderOrganizations**?{*param*}

Get the list of senderOrganizations, possibly filtered with different criteria.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| senderApplicationId  [string] | query-param | Filter only senderOrganizations that are linked to the senderApplication represented by the given applicationId. |
| messageTypeId  [string] | query-param | Filter only senderOrganizations that can publish messages of the messageType represented by the given messageTypeId. |
| sort  [string]  *[default: sort=organizationId]* | query-param | Comma-separated list of properties to sort data. The default sorting direction is ascending. Names may be prefixed with ‘-‘ to sort descending.  ?sort=-organizationShortNameFr |
| page [number]  *[default: page=1]* | query-param | Use this parameter to request a specific page. Page numbers are 1-based.  ?page=3 |
| pageSize [number]  *[default: pageSize=25]* | query-param | Use this parameter to specify the page size.  ?pageSize=12 |
| Response | | |
| body | The response is a **collection** JSON object. Each item corresponds to the resource type “senderOrganizationSummary”.  A **senderOrganizationSummary** is composed of the following attributes:   * organizationId [**required**] [string] * organizationShortName [**required**] [translatedString] * organizationLogo [**required**] [list images] * messageTypeIds [0..\*] [**optional**] [list] * senderApplicationIds [0..\*] [**optional**] [list]   **sort by default of collection:**  ***+*** *companyId (ascending),* ***+*** *applicationId (ascending)* | {  "items": [  {  "organizationId": "0123456789",  "organizationShortName": {  "nl": "FPD",  "fr": "SFP",  "de": "FPD",  "en": "FPS"  },  "organizationLogo": {  "items": [{  "imageId": "logo1",  "size": 200,  "format": "image/png",  "language": "nl",  "content": "$base64 data",  "href": "../../images/logo1"  }],  "totalItems": 1  },  "href": "./senderOrganizations/0123456789"  },  {  "organizationId": "0206731645",  "organizationShortName": {  "nl": "RSZ",  "fr": "ONSS",  "de": "LSS",  "en": "NSSO"  },  "organizationLogo": {  "items": [{  "imageId": "logoOnss",  "size": 500,  "format": "image/png",  "language": "fr",  "content": "$base64 data",  "href": "../../images/logoOnss"  },  {  "imageId": "logoRsz",  "size": 500,  "format": "image/png",  "language": "nl",  "content": "$base64 data",  "href": "../../images/logoRsz"  },{  "imageId": "logoLss",  "size": 500,  "format": "image/png",  "language": "nl",  "content": "$base64 data",  "href": "../../images/logoLss"  }],  "totalItems": 3  },  "href": "./senderOrganizations/0206731645"  }  ],  "totalItems": 2  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### GET referenceData/senderOrganizations**/{$organizationId}**

Get the details about a given senderOrganization, identified by its *organizationId*.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| organizationId  [string] | path-param | CompanyID of the sender organization. |
| Response | | |
| body | A **senderOrganization** is composed of the following attributes:   * organizationId [**required**] [string] * organizationShortName [**required**] [translatedString] * organizationLongName [**optional**] [translatedString] * organizationLogo [**optional**] [list images] * organizationUrl [**optional**] [translatedString] * senderApplicationIds [0..\*] [**optional**] [list] * messageTypeIds [0..\*] [**optional**] [list] * contactBusiness [**optional**] [string] * contactTechnical [**optional**] [string] | {  "organizationId": "0123456789",  "organizationShortName": {  "nl": "FPD",  "fr": "SFP",  "de": "FPD",  "en": "FPS"  },  "organizationLongName": {  "nl": "Federale Pensioendienst",  "fr": "Service fédéral des Pensions",  "de": "Föderaler Pensionsdienst",  "en": "Federal Pension Service"  },  "organizationLogo": {  "items": [{  "imageId": "logo1",  "size": 200,  "format": "image/png",  "language": "nl",  "content": "$base64 data",  "href": "../../images/logo1"  }],  "totalItems": 1  },  "organizationUrl": {  "nl": "http://www.onprvp.fgov.be/NL/profes/Paginas/default.aspx",  "fr": "http://www.onprvp.fgov.be/fr/profes/pages/default.aspx",  "de": "http://www.onprvp.fgov.be/de/profes/seiten/default.aspx"  },  "senderApplicationIds": [  "infirmarySystem"  ],  "messageTypeIds": [  "PatientProof"  ],  "contactBusiness": "Eranova@rszonsslss.be",  "contactTechnical": "eboxintegration@smals.be"  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### GET referenceData**/senderApplications**?{*param*}

Get the list of senderApplications, possibly filtered with different criteria.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| senderOrganizationId  [string] | query-param | Filter only senderApplications that are linked to the senderOrganization represented by the given organizationId. |
| messageTypeId  [string] | query-param | Filter only senderApplications that can publish messages of the messageType represented by the given messageTypeId. |
| name  [string] | query-param | Filter only applications that contain a specific string in the name, description or identifier. |
| sort  [string]  *[default: sort=applicationId]* | query-param | Comma-separated list of properties to sort data. The default sorting direction is ascending. Names may be prefixed with ‘-‘ to sort descending.  ?sort=-applicationNameFr |
| page [number]  *[default: page=1]* | query-param | Use this parameter to request a specific page. Page numbers are 1-based.  ?page=3 |
| pageSize [number]  *[default: pageSize=25]* | query-param | Use this parameter to specify the page size.  ?pageSize=12 |
| Response | | |
| body | The response is a **collection** JSON object. Each item corresponds to the resource type “senderApplicationSummary”.  A **senderApplicationSummary** is composed of the following attributes:   * applicationId [**required**] [string] * applicationName [**required**] [translatedString] * senderOrganizationIds [0..\*] [**optional**] [list] * messageTypeIds [0..\*] [**optional**] [list]   **sort by default of collection:**  ***+*** *organizationId (ascending),* ***+*** *applicationId (ascending)* | {  "items": [{  "applicationId": "mypension:citizen",  "applicationName": {  "nl": "Pensioenkadaster",  "fr": "Kadastre de pension",  "de": "Kadastre de pension",  "en": "Pensioenkadaster"  },  "senderOrganizationIds": [  "0123456789"  ],  "messageTypeIds": [  "PensionAttest"  ],  "href": "./senderApplications/mypension:citizen"  },  {  "applicationId": "infirmarySystem",  "applicationName": {  "nl": "Verpleegkundig systeem",  "fr": "Système de soins infirmiers",  "de": "Système de soins infirmiers",  "en": "Infirmary system"  },  "senderOrganizationIds": [  "0206731645"  ],  "messageTypeIds": [  "PatientProof"  ],  "href": "./senderApplications/infirmarySystem"  }  ],  "totalItems": 2  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### GET referenceData/senderApplications**/{$senderApplicationId}**

Get the details about a given senderApplication, identified by its *senderApplicationId*.

|  |  |  |
| --- | --- | --- |
| ​​​Parameters | | |
| senderOrganizationId [string] | path-param | CompanyID of the sender organization. |
| Response | | |
| body | A **senderApplication** is composed of the following attributes:   * applicationId [**required**] [string] * applicationName [**required**] [translatedString] * applicationDescription [**optional**] [translatedString] * applicationLogo [**optional**] [list images] * applicationUrl [**optional**] [translatedString] * messageTypeIds [0..\*] [**optional**] [list] * senderOrganizationIds [0..\*] [**optional**] [list] * contactBusiness [**optional**] [string] * contactTechnical [**optional**] [string] | {  "applicationId": "infirmarySystem",  "applicationName": {  "nl": "Verpleegkundig systeem",  "fr": "Système de soins infirmiers",  "de": "Système de soins infirmiers",  "en": "Infirmary system"  },  "applicationDescription": {  "nl": "Beschrijving",  "fr": "Description",  "de": "Beschreibung",  "en": "Description"  },  "applicationLogo": {  "items": [{  "imageId": "applicationLogo1",  "size": 200,  "format": "image/png",  "language": "nl",  "content": "$base64 data",  "href": "../../images/applicationLogo1"  }],  "totalItems": 1  },  "senderOrganizationIds": [  "0206731645"  ],  "messageTypeIds": [  "PatientProof"  ],  "contactBusiness": "eranova@rszonsslss.be",  "contactTechnical": "eboxintegration@smals.be"  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

# Out-of-scope (MVP)

* All operations different than a GET on the identified resources.
* Support multiple rendering possibilities, returning different amounts of information concerning a resource based on the chosen rendering type.
* Use scenarios of the REST Service in which the identification of the citizen (ebox) cannot be deduced from the AccessToken, and must therefore be provided in a different way (e.g. parameter).
* Management of the CitizenProfile
  + E.g. get the contact data of a given citizen
* Use scenarios for the NotificationEngine
  + E.g. which citizens have to be notified? When should they be notified? And about what? …
* Payment lifecycle
  + E.g. validation/status/… about a payment
* Signed attachment verification support
  + E.g. Verifying CAdES signed document in DETACHED mode
* Encryption/decryption support for messages or attachments
* Details about messages provided as ‘registered mail’
  + E.g. proof of sending (from the third party delivery system or Certification service provider), acknowledgment of receipt…

# Standard REST Conventions and common elements

In this section, we resume the main standardized conventions and common elements used to design the RESTful API.

## Resource URI

The URI notation is in **lowerCamelCase** to enhance readability and to separate compound names.

The URI **never contains a file extension**.

## Reserved words - Query parameters

The query component of a URI contains a set of parameters to be interpreted as a variation or derivative of the resource. The query component can provide clients with additional interaction capabilities such as ad hoc searching and filtering.

|  |  |  |  |
| --- | --- | --- | --- |
| **Term** | **Description** | **Example** | **Ref** |
| lang | Requests to return the information in a specific language. The language code used is the [two-letter ISO 639-1 standard](http://www.loc.gov/standards/iso639-2/php/code_list.php) | ?lang=nl |  |
| page | When a collection resources is paged, use this parameter to request a specific page. Page numbers are 1-based. | ?page=3&pageSize=20 | Paging over a large collection​ |
| pageSize | When a collection resources is paged, use this parameter to specify the page size. | ?page=3&pageSize=20 | Paging over a large collection​ |
| q | The standard search parameter to do a full-text search. | ?q=pension | Filtered collection |
| sort | Comma-separated list of properties to sort data.  To indicate sorting direction, names may be prefixed with + (ascending, default) or - (descending) | ?sort=+name |  |

The query-param **q** is reserved to implement a full text search on all the resource’s content. ​

*Example Query parameters*

​ REST/ebox-demo/v1/senderOrganizations**?q=sigedis**  
Filter the resource collection to results

REST/ebox-demo/v1/senderOrganizations/0880820673​**?fields=organizationShortName,organizationLogo** Filter the resource properties to the ones specified in the fields query parameter.

REST/ebox-demo/v1/senderOrganizations/0880820673**?lang=nl**  
Only return translatable properties in dutch.

### JSON properties

All JSON property names are in English and are written in lower camel case notation. Do not use underscore or hyphens for composite words (Properties from the [HAL standard](http://integrationsoa/rest/#hal-links) are an exception to this rule).

* Properties with a <null> value should be stripped from the JSON message.
* Empty JSON arrays are never stripped from the JSON message.
* The JSON properties have no specific order inside a JSON object.
* Dates are written in ISO 8601 full-date format: yyyy-mm-dd
  + See [Swagger Datatypes](http://swagger.io/specification/#data-types-12) and [RFC 3339 section 5.6](https://tools.ietf.org/html/rfc3339#section-5.6).
* Date/time are written in ISO 8601 date-time format: yyyy-MM-dd’T’HH:mm:ssXXX
  + See [JSON Schema Validation 8.3.1. date-time](http://json-schema.org/latest/json-schema-validation.html#rfc.section.8.3.1) and [RFC 3339 section 5.6](https://tools.ietf.org/html/rfc3339#section-5.6).

## Link references

Hypertext Application Language (HAL) is a simple format that gives a consistent and easy way to hyperlink between resources in your API. (<https://tools.ietf.org/html/draft-kelly-json-hal-08>)

Each JSON object has a predefined \_links property.

| **Term** | **Description** | **Example** |
| --- | --- | --- |
| \_links | JSON property on root level grouping all hyperlinks from this resource. | "\_links": {  "self": {  "href": "/messages/*$messageId*",  "type": "message"  }  } |
| next | The next-reference contains the absolute URL of the next page in a paged collection. | "\_links": {  "next": {  "href": "/messages?page=3&pageSize=2"  }  } |
| previous | The previous-reference contains the absolute URL of the previous page in a paged collection. | "\_links": {  "previous": {  "href": "/messages?page=3&pageSize=2"  }  } |
| self | The self-reference contains the absolute URL of the resource itself. | "\_links": {  "self": {  "href": "/messages/*$messageId*",  "type": "message"  }  } |

## Collection

Each collection resource can decide to return

1. either an **array of resource links** (potentially extended with some business properties to show a summary)
2. or **an array of the resource itself** (useful when the business concept is simple / contains little data)

### List of resources links

A query on this kind of collection returns links to the resources in its result. The link object can be extended with some additional business properties, needed to display in a master view.

|  |  |  |
| --- | --- | --- |
| ​​​​​​​​​[GET](http://integrationsoa/rest/index.html#get) | /​​messageTypes | get all the messageTypes in the collection |
| ​​​Parameters | | |
| Response | | |
| body | The response is a collection JSON object.   * **items** is the reserved word for the array containing the result. * **totalItems** is the reserved word for all the items in the collection result.   Each item should contain a JSON link object to a resource document.   * **href** contains a link to the resource document. * **title** contains a human understandable text to describe the resource * **type** contains the JSON object class of the resource | ​​​{  "items":[  {  "href":"[/messageType/93017373](http://rest-reference.test.paas.socialsecurity.be/REST/demo/v1/employers/93017373)",  "title":"@todo",  "messageTypeId": "@todo",  "type":"messageType"  },  {  "href":"[/messageType/12978696](http://rest-reference.test.paas.socialsecurity.be/REST/demo/v1/employers/12978696)",  "title":"@todo",  "type":"messageType",  " messageTypeId ": "@todo",  },  {  "href":"[/messageType/20620259](http://rest-reference.test.paas.socialsecurity.be/REST/demo/v1/employers/20620259)",  "type":"messageType",  "messageTypeId": "@todo",  }  ],  "totalItems":3  }​ |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the collection is empty ​ |

### List of resources

A query on this kind of collection returns the resources themselves in its result. This is useful when the resource contains little data and to avoid navigating to the resource document.

|  |  |  |
| --- | --- | --- |
| ​​​​​​​​​[GET](http://integrationsoa/rest/index.html#get) | ebox/messages/{*$messageId*}/attachments | get all the attachments for a given *messageId*  NB: |
| Response | | |
| body | The response is a collection JSON object.   * **items** is the reserved word for the array containing the result. * **totalItems** is the reserved word for all the items in the collection result.   Each item is the same JSON object as the document resource.   * **\_links/self** contains the link to the resource document. | ​​​{  "items": [  {  **"attachmentId": "a6006ad3dba0"**,  "attachmentTitle": {  "nl": "attest.pdf",  "fr": "attestation.pdf",  "de": "attest.pdf",  "en": "attest.pdf"  },  "attachmentSigned": false,  "mediaType": "application/pdf",  "digest": {  "digestValue": "szOe0ZbpGXDLMT9vs1dsoJLM0hgkVCcrQyAJpfFt6FtwQTl5wPdTVZfdoKh73/u7ZFiFgUHE15LqMvhV6BLCLg==",  "digestMethod": "SHA-512"  },  "size": "45",  "\_links":  {  "self": {"href": "ebox/messages/8a35f632-7b4e-11e7-bb31-be2e44b06b34/attachments/a6006ad3dba0"}  }  },  {  **"attachmentId": "0339ZEIFSDIOAC"**,  "attachmentTitle": {  "nl": "details.csv",  "fr": "details.csv”  },  "mediaType": "text/csv",  "size": "190",  "\_links":  {  "self": {"href": "ebox/messages/8a35f632-7b4e-11e7-bb31-be2e44b06b34/attachments/0339ZEIFSDIOAC"}  }  }  ],  "totalItems":2  }​ |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the collection is empty ​ |

### Filtered collection

A collection can be filtered using query parameters.​ You can filter on a specific resource property by specifying the property name as query param.

|  |  |  |
| --- | --- | --- |
| [GET](http://integrationsoa/rest/index.html#get) | /messageTypes | get all the messageTypes in the collection |
| ​​​Parameters | | |
| senderOrganizationId | query-param | Filter only messageTypes that are related to a given *senderOrganizationId e.g:* ?senderOrganizationId=0206737484 |
| Response | | |
| body |  | ​​​{  "items": [  {  "messageTypeId": *"OnemRvaFiscalIndexCard "*,  "messageTypeName": {  "nl": *"Tijdskredit attest"*,  "fr": *"Attestation Crédit-temps"*,  "de": *"Zeitkredit-Bescheinigung"*,  },  **"senderOrganizationId": *“0206737484"*,**  "href": *"/messagesTypes/OnemRvaFiscalIndexCard/"*  },  {  "messageTypeId": *"OnemRvaDecisionC62"*,  "messageTypeName": {  "nl": "*Beslissing C62*",  "fr": "*Décision C62"*  },  **"senderOrganizationId": *“0206737484"*,**  "href": *"/messagesTypes/OnemRvaDecisionC62/"*  }  ],  "totalItems": 2,  "\_links": {}  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

### Paging over a large collection

When a collection contains too many results, the results should be paged. ​

|  |  |  |
| --- | --- | --- |
| ​​​​​​​​​[GET](http://integrationsoa/rest/index.html#get) | /messages | get all the messages in the collection |
| ​​​Parameters | | |
| ​​**page** | query-param | The index of the current page of items. It should be 1-based (the default and first page is 1). |
| ​**pageSize** | query-param | The maximal number of results per page. |
| Response (*example*): | | |
| ​​​​​​​​​[GET](http://integrationsoa/rest/index.html#get) | /messages?**page=2**&**pageSize=2** | get the 2nd page of messages in the collection, with a pageSize of 2 |
| body | ​The response is a collection JSON object.   * the relation **next** is reserved word for the hyperlink to the next page * the relation **previous** is reserved word for the hyperlink to the previous page | ​{  "items": [  {  "messages": {  "items": [{  **"messageId": "9D000003NG00U",**  "subject": { "nl": "Aanvraag pensioengegevens",  "fr": "Demande données de pension"  },  "receiptDate": "2017-05-20T10:15:00Z",  "readStatus": false,  "messageTypeId": "PensionAttest",  "senderOrganizationId": "0123456789",  "href": "/messages/9D000003NG00U"  },  {  **"messageId": "16cd9678-5687-11e7",**  "subject": {"nl": "Horeca@work attest"},  "receiptDate": "2017-05-24T09:26:00Z",  "readStatus": true,  "messageTypeId": "HorecaWork",  "senderOrganizationId": "0206731645",  "senderApplicationId": "repository: relationship:employer-worker:horeca-at-work",  "href": "/messages/16cd9678-5687-11e7"  }  ],  "totalItems": 7,  "\_links": {  "**next**": {"href": "[/messages?page=3&pageSize=2](http://rest-reference.test.paas.socialsecurity.be/REST/demo/v1/companies?page=3&pageSize=2)"},  "**previous**": {href":"[/messages?page=1&pageSize=2](http://rest-reference.test.paas.socialsecurity.be/REST/demo/v1/companies?page=1&pageSize=2)"}  }  } |
| Response codes ​​ | | |
| [200](http://integrationsoa/rest/index.html#http-200) | OK | Default response code, also when the filtered collection is empty ​ |

# Common Status codes

The full list of HTTP status codes is documented [here](http://www.ietf.org/assignments/http-status-codes/http-status-codes.xml).

In order to conform to the REST uniform service contract, REST services should stick to this reduced list of status codes. The addition of new status codes should be evaluated by the REST policy group.

## [1xx Informational](http://integrationsoa/rest/#1xx-informational)

Request received, continuing process

## [2xx Success](http://integrationsoa/rest/#2xx-success)

The action was successfully received, understood, and accepted

| **Code** | **Description** | **Operations** |
| --- | --- | --- |
| 200 OK | **200 (“OK”) should be used to indicate nonspecific success**  In most cases, 200 is the code the client hopes to see. It indicates that the REST API successfully carried out whatever action the client requested, and that no more specific code in the 2xx series is appropriate. Unlike the 204 status code, a 200 response should include a response body. | [GET](http://integrationsoa/rest/#get), [PUT](http://integrationsoa/rest/#put) |
| 201 Created | **201 (“Created”) must be used to indicate successful resource creation**  A REST API responds with the 201 status code whenever a collection creates, or a store adds, a new resource at the client’s request. There may also be times when a new resource is created as a result of some controller action, in which case 201 would also be an appropriate response. | [PUT](http://integrationsoa/rest/#put) |
| 202 Accepted | **202 (“Accepted”) must be used to indicate successful start of an asynchronous action**  A 202 response indicates that the client’s request will be handled asynchronously. This response status code tells the client that the request appears valid, but it still may have problems once it’s finally processed. A 202 response is typically used for actions that take a long while to process. Controller resources may send 202 responses, but other resource types should not. |  |
| 204 No Content | **204 (“No Content”) should be used when the response body is intentionally empty**  The 204 status code is usually sent out in response to a PUT, POST, or DELETE request, when the REST API declines to send back any status message or representation in the response message’s body. An API may also send 204 in conjunction with a GET request to indicate that the requested resource exists, but has no state representation to include in the body. | [GET](http://integrationsoa/rest/#get), [PUT](http://integrationsoa/rest/#put), [POST](http://integrationsoa/rest/#post), [DELETE](http://integrationsoa/rest/#delete) |

## [3xx Redirection](http://integrationsoa/rest/#3xx-redirection)

Further action must be taken in order to complete the request|

| **Code** | **Description** | **Operations** |
| --- | --- | --- |
| 301 Moved Permanently | **301 (“Moved Permanently”) should be used to relocate resources**  The 301 status code indicates that the REST API’s resource model has been significantly redesigned and a new permanent URI has been assigned to the client’s requested resource. The REST API should specify the new URI in the response’s Location header. |  |
| 303 See Other | **303 (“See Other”) should be used to refer the client to a different URI**  A 303 response indicates that a controller resource has finished its work, but instead of sending a potentially unwanted response body, it sends the client the URI of a response resource. This can be the URI of a temporary status message, or the URI to some already existing, more permanent,resource.  Generally speaking, the 303 status code allows a REST API to send a reference to a resource without forcing the client to download its state. Instead, the client may send a GET request to the value of the Location header. |  |
| 304 Not Modified | **304 (“Not Modified”) should be used to preserve bandwidth**  This status code is similar to 204 (“No Content”) in that the response body must be empty. The key distinction is that 204 is used when there is nothing to send in the body, whereas 304 is used when there is state information associated with a resource but the client already has the most recent version of the representation. This status code is used in conjunction with conditional HTTP requests |  |
| 307 Temporary Redirect | **307 (“Temporary Redirect”) should be used to tell clients to resubmit the request to another URI**  A 307 response indicates that the REST API is not going to process the client’s request. Instead, the client should resubmit the request to the URI specified by the response message’s Location header.  A REST API can use this status code to assign a temporary URI to the client’s requested resource. For example, a 307 response can be used to shift a client request over to another host. |  |

## [4xx Client Error](http://integrationsoa/rest/#4xx-client-error)

The request contains bad syntax or cannot be fulfilled.

| **Code** | **Description** | **Operations** |
| --- | --- | --- |
| 400 Bad Request | **400 (“Bad Request”) may be used to indicate nonspecific failure** 400 is the generic client-side error status, used when no other 4xx error code is appropriate. |  |
| 401 Unauthorized | **401 (“Unauthorized”) must be used when there is a problem with the client’s credentials.**  A 401 error response indicates that the client tried to operate on a protected resource without providing the proper authorization. It may have provided the wrong credentials or none at all. |  |
| 403 Forbidden | **403 (“Forbidden”) should be used to forbid access regardless of authorization state.**  A 403 error response indicates that the client’s request is formed correctly, but the REST API refuses to honor it. A 403 response is not a case of insufficient client credentials; that would be 401 (“Unauthorized”). |  |
| 404 Not Found | **404 (“Not Found”) must be used when a client’s URI cannot be mapped to a resource.**  The 404 error status code indicates that the REST API can’t map the client’s URI to a resource. |  |
| 405 Method Not Allowed | **405 (“Method Not Allowed”) must be used when the HTTP method is not supported.**  The API responds with a 405 error to indicate that the client tried to use an HTTP method that the resource does not allow. For instance, a read-only resource could support only GET and HEAD, while a controller resource might allow GET and POST, but not PUT or DELETE.  A 405 response must include the Allow header, which lists the HTTP methods that the resource supports. For example: Allow: GET, POST |  |
| 406 Not Acceptable | **406 (“Not Acceptable”) must be used when the requested media type cannot be served**  The 406 error response indicates that the API is not able to generate any of the client’s preferred media types, as indicated by the Accept request header. For example, a client request for data formatted as application/xml will receive a 406 response if the API is only willing to format data as application/json. |  |
| 409 Conflict | **409 (“Conflict”) should be used to indicate a violation of resource state**  The 409 error response tells the client that they tried to PUT the REST API’s resources into an impossible or inconsistent state. For example, a REST API may return this response code when a client tries to DELETE a non-empty store resource. | [PUT](http://integrationsoa/rest/#put),[POST](http://integrationsoa/rest/#post),[PATCH](http://integrationsoa/rest/#patch) |
| 412 Precondition Failed | **412 (“Precondition Failed”) should be used to support conditional operations**  The 412 error response indicates that the client specified one or more preconditions in its request headers, effectively telling the REST API to carry out its request only if certain conditions were met. A 412 response indicates that those conditions were not met, so instead of carrying out the request, the API sends this status code.  **Only use for**[**conditional HTTP requests**](http://tools.ietf.org/html/rfc7232)**, not constraints expressed in the HTTP payload.**Use [409 Conflict](http://integrationsoa/rest/#http-409) instead. |  |
| 415 Unsupported Media Type | **415 (“Unsupported Media Type”) must be used when the media type of a request’s payload cannot be processed**  The 415 error response indicates that the API is not able to process the client’s supplied media type, as indicated by the Content-Type request header. For example, a client request including data formatted as application/xml will receive a 415 response if the API is only willing to process data formatted as application/json. |  |

## [5xx Server Error](http://integrationsoa/rest/#5xx-server-error)

The server failed to fulfill an apparently valid request

| **Code** | **Description** | **Operations** |
| --- | --- | --- |
| 500 Internal Server Error | **500 (“Internal Server Error”) should be used to indicate API malfunction**  500 is the generic REST API error response. Most web frameworks automatically respond with this response status code whenever they execute some request handler code that raises an exception.  A 500 error is never the client’s fault and therefore it is reasonable for the client to retry the exact same request that triggered this response, and hope to GET a different response. |  |
| 503 Service Unavailable | **503 (“Service Unavailable”) indicates that the server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay.**  The server MAY send a Retry-After header field to suggest an appropriate amount of time for the client to wait before retrying the request. |  |

# Specific Error Codes

In the following table, we describe the specific error codes for the citizen federated-ebox.

| **Code** | **Description** | **Operations** |
| --- | --- | --- |
| FEDBOX-001 | INVALID\_TOKEN: Access token is not valid (unsupported or rejected by the Authorization Server) | GET |
| FEDBOX-002 | EXPIRED\_TOKEN: Access token is expired and thus not valid anymore | GET |
| FEDBOX-003 | INSUFFICIENT\_SCOPE: Access token is not sufficient to use this operation (the provided token does not cover the scope of the called operation).  NB: not specific to a resource; see also FEDBOX-014. | GET |
| FEDBOX-010 | NOT\_IMPLEMENTED: the given *$paramName* is not yet implemented in this service.  NB: when *paramName* is supported by the federated API.  (see also the *Provider Registry* to know which functionalities are implemented or not) | GET /resources*?param* |
| FEDBOX-011 | INVALID\_PARAM\_NAME: the given ***$paramName*** is not a valid query-param. | GET /resources*?param* |
| FEDBOX-012 | INVALID\_PARAM\_VALUE: the value provided for the ***$paramName*** query-param is not valid.   * NB1: the format of an invalid query-param SHOULD be specified by the service in the error message.   + **Example:** INVALID\_PARAM\_VALUE: the value provided for the ***pageSize*** query-param is not valid. Excepted value must be **an integer (int32).** * NB2: if the excepted value of a query-param is defined in an enumeration, the error message SHOULD precise this enumeration list.   + **example:** INVALID\_PARAM\_VALUE: the value provided for the ***sort*** query-param is not valid. Excepted values must be in ***{+receiptDate, -receiptDate***} * NB3: if the given query-parameter(s) is (are) syntactically correct but does not result in any elements found, the service MUST NOT return an error code. The expected result is here an empty list, with ‘totalItems=0’.   (? Quid si plusieurs param. Invalides: ok si le service retourne seulement la 1ère erreur trouvée ?) | GET /resources*?param* |
| FEDBOX-013 | NOT\_FOUND: the resource you tried to access is not found for the given ***$paramName.***  **Example:** NOT\_FOUND: the resource you tried to access is not found for the given *messageId*. | GET /resources  */{resourceId}* |
| FEDBOX-014 | NOT\_AUTHORIZED: the resource you tried to access for the given ***$paramName*** is not authorized for the Access Token used.  **Example:** NOT\_AUTHORIZED: the resource you tried to access for the given *labelId* is not authorized for the Access Token used. | GET /resources  */{resourceId}* |
| FEDBOX-020 | BLOCKED\_EBOX: the ebox corresponding to this citizen (deducted from the Access Token) is blocked. | GET /ebox |
| FEDBOX-021 | ? cas d’un Provider qui bloque l’appelant (IP-adress)… | GET /ebox/… |
| FEDBOX-030 | NOT\_AVAILABLE: this operation is temporary not available.  See also FEDBOX-010: NOT\_IMPLEMENTED  http: [503 Service Unavailable](http://integrationsoa.smals.be/rest/#http-503) |  |
| FEDBOX-031 | UNEXPECTED\_ERROR: An unexpected error has occurred.  Retries should work. If the problem persists service desk may help.  http: 500 Internal Server Error |  |

## Error handling : complete examples

|  |  |  |
| --- | --- | --- |
| **Complete example 1 :** *specific resource not found* | HTTP/1.1 **404 Not Found**  Server: Apache-Coyote/1.1  Content-Type: application/json  {  "id":"d9e35127-e9b1-4201-a211-2b52e52508df",  "code": "**FEDBOX-013**",  "message": "**NOT\_FOUND**",  "details": [{  "kind": "**path-param**",  "message": "**NOT\_FOUND: the resource you tried to access is not found for the given messageId.**",  "ref": "**messageId**",  "value": "**9D00008RKF006**"  }]  } |  |
| **Complete example 2 :** *invalid query-param to filter a list of resources* | HTTP/1.1 **400 Bad Request**  Server: Apache-Coyote/1.1  Content-Type: application/json  {  "id": "88d8393b-f314-4cb3-8ef8-047971dba4ba",  "code": "**FEDBOX-012**",  "message": "**INVALID\_PARAM\_VALUE**",  "details": [{  "kind": "**query-param**",  "message": "**INVALID\_PARAM\_VALUE: the value provided for the pageSize query-param is not valid. Excepted value must be an integer (int32.**",  "ref": "**pageSize**",  "value": "**large**"  }]  } |  |

1. It is necessary to make the difference between **document senders** and **MessageRegistry Providers**. A sender posts a message to a citizen ebox, by means of a publishing service provided by a MessageRegistry Provider. The MessageRegistry Provider stores the message on its infrastructure, and provides a RESTful consultation service (*aka* Message Registry, based on this Message Registry Contract). This service will be finally integrated into the federated view that will be able to gather all the ebox data of a citizen. [↑](#footnote-ref-2)