Contents

Versions history.............................................................................................................3
Introduction .........................................................................................................................4
Discover e-Box Enterprise (user view)..............................................................................4
The federated model of the Enterprise e-Box (technical view).........................................5
   Diagram: high-level technical view ..............................................................................5
Glossary .............................................................................................................................6
Discover the RESTful ebox/enterprise API .....................................................................7
   Diagram: API discovering – access to ebox resources .................................................7
DocProvider Onboarding process....................................................................................8
  1. DocProvider initial request .......................................................................................8
  2. Agreement in principle of the NSSO .........................................................................8
  3. Kick-off meeting.........................................................................................................8
  4. Formal request & contract + certificate ....................................................................8
  5. Request validation & configuration (ACC) ...............................................................9
  6. Implementation and tests ..........................................................................................9
  7. Integration of your WS & configuration (PRD) .........................................................9
  8. Deploy in Production ................................................................................................9
  9. Production-ready .......................................................................................................9
DocProvider technical integration ..................................................................................9
   Points of attention for DocProvider Integration .......................................................9
Annexes ..........................................................................................................................13
Annex 1: X.509 certificate..............................................................................................13
Annex 2: OAuth2 Authorization Server onboarding .....................................................14
   Token validation & analysis........................................................................................14
   Reminder about identity..............................................................................................14
   Reminder resources access control ..........................................................................14
   Example.........................................................................................................................15
Annex 3: References ......................................................................................................15
## Versions history

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Author</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Draft</td>
<td>2018-05</td>
<td>Antoine MOULART</td>
<td>First draft</td>
</tr>
<tr>
<td>0.2</td>
<td>Draft</td>
<td>2018-05-14</td>
<td>Antoine MOULART</td>
<td>Peer-review avec Kevin NOPPE</td>
</tr>
<tr>
<td>0.3</td>
<td>Draft</td>
<td>2018-06</td>
<td>Antoine MOULART</td>
<td>English Translation</td>
</tr>
<tr>
<td>0.4</td>
<td>Draft</td>
<td>2018-08-03</td>
<td>Antoine MOULART</td>
<td>Review before final proposal</td>
</tr>
<tr>
<td>0.5</td>
<td>Draft</td>
<td>2018-08</td>
<td>Antoine MOULART</td>
<td>Final first proposal</td>
</tr>
<tr>
<td>0.6</td>
<td>Draft</td>
<td>2018-08-21</td>
<td>Antoine MOULART</td>
<td>Add points of attention for technical integrations</td>
</tr>
<tr>
<td>0.7</td>
<td>Draft</td>
<td>2018-09</td>
<td>Antoine MOULART</td>
<td>Split onboarding processes + update after internal review</td>
</tr>
<tr>
<td>0.8</td>
<td>Proposal</td>
<td>2018-09-06</td>
<td>Antoine MOULART</td>
<td>Separated version for DocProvider + OAuth2 integration info from THIM ANNEESSENS</td>
</tr>
</tbody>
</table>
Introduction

This e-Box Enterprise cookbook is intended for technical teams who, on behalf of their organization (user company, Institution, Partner platform...) are in charge of a technical integration with the e-Box system.

Several types of integration are possible; this cookbook will focus on the e-Box DocProvider integration.

When an aspect is already described elsewhere, this cookbook will not duplicate the information, but will indicate where exactly to find it.

Discover e-Box Enterprise (user view)

The e-Box Enterprise is the secure electronic mailbox, available for free for any company in Belgium. The classic user-interface of the e-Box Enterprise takes the form of a Java web-application:

The user help for the web interface of the e-Box Enterprise is not the subject of this cookbook. For documentation about the classic web application, concerning the activation of the e-Box and the features, we refer you to the links below:

- Access to e-Box Enterprise on the Social Security Portal:
The federated model of the Enterprise e-Box (technical view)

The diagram below shows the high-level technical view of the federated model of the e-Box Enterprise. The federated view indeed integrates several message sources (DocProviders), each accessible via a MessageRegistry (WS REST) respecting the same contract (ebox / common enterprise API).

In addition, some institutions (DocSenders) can continue to publish their messages to a DocProvider who will make them available for the federated e-Box.

Finally, a company (DocConsumer) can directly retrieve the content of its e-Box via a technical integration with each MessageRegistry offering this type of access. Such integration is interesting for a (large) company that does not want to use the traditional user-interface, and wants to automate the unloading and management of its e-Box in its internal system.

See also Glossary below.

Diagram: high-level technical view
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td>Application Programming Interface, of the RESTful type in this model, which is defined as a common contract for all actors of the e-Box system.</td>
</tr>
<tr>
<td><strong>Authorization Server</strong></td>
<td>Authorization server that delivers the Tokens to the client (DocConsumer). These Tokens will then be used in order to call the different Message Registries.</td>
</tr>
<tr>
<td><strong>DocConsumer</strong></td>
<td>Enterprise (private/public company) that retrieves the content of its own e-Box through a backend integration (WS call) rather than consulting the web application in a conventional way.</td>
</tr>
<tr>
<td><strong>DocProvider</strong></td>
<td>Organization that hosts its documents on its own infrastructure, and makes them accessible to the Enterprise e-Box, by implementing its MessageRegistry (WS REST), that respects the common API and the required security standards.</td>
</tr>
<tr>
<td><strong>DocSender</strong></td>
<td>Organization that publishes documents in a classic way to the e-Box service of the Social Security, and lets the responsibility to this DocProvider itself of storing and securing access to documents. A DocSender is in practice identified by the BCE / KBO number of its organization, and an application identifier (URN), which is typically an online service for businesses (e.g. Sigedis - DB2P application).</td>
</tr>
<tr>
<td><strong>Message-Registry</strong></td>
<td>Web service allowing access to the messages of a DocProvider. Each RESTful WS must respect the common API, and support the defined security standards.</td>
</tr>
<tr>
<td><strong>META-API</strong></td>
<td>The META-API corresponds to several 'technical' operations that each DocProvider can use to describe the degree of implementation of the 'business' operations, and indicate which elements are implemented or not (example: sorting or optional query-param ...).</td>
</tr>
<tr>
<td><strong>Provider-Registry</strong></td>
<td>Technical web service that lists the different DocProviders available, with the URLs of each MessageRegistry. The ProviderRegistry is therefore seen as the directory of all accessible message sources.</td>
</tr>
</tbody>
</table>
Discover the RESTful ebox/enterprise API

The RESTful ebox/enterprise API is documented via the OpenAPI v2 Specification (OAS) [aka Swagger] standard. This specification is available in the .yaml format. In addition to the Swagger, the resource description is also detailed in the form of JSON-Schemas, in .json format.

The ebox/enterprise API is completed by a META-API that specifically describes the optional elements of the API, so that each DocProvider can indicate whether or not these elements are implemented in its MessageRegistry.

Public versions of the API and JSON-Schemas are available through the EBOX Members Area: https://members.intranext.smals.be/projects/EBOX/default.aspx

There is also published a presentation (.pptx format) of the API with different examples and diagrams to better visualize access to resources.
DocProvider Onboarding process

1. **DocProvider initial request**
   First you can contact eBoxIntegration@smals.be in order to explain in a high-level way what is expected with you future e-Box integration as new DocProvider, for which business/use cases, and a first idea of the volume (number of provided documents) and the expected planning.

2. **Agreement in principle of the NSSO**
   Before starting any integration, it is necessary to obtain a first agreement in principle from the NSSO. This agreement will be requested by the eBoxIntegration team to the NSSO e-Box managers.

3. **Kick-off meeting**
   This step is optional but often very useful in order to answer any questions before starting the technical integration.

4. **Formal request & contract + certificate**
In order to officially become a new DocProvider, please send the following documents to eBoxIntegration@smals.be:

- e-Box DocProvider onboarding form & contract (the contract must be signed by a Legal Representative from your organization);
- The public part of your certificate;

If you need to order a X.509 certificate, please pay attention to respect the expected format. See Annex 2 to have all the details about that. Once completed, the form must be sent.

5. Request validation & configuration (ACC)

The eBoxIntegration team is responsible for technically validating the received form. A formal validation of the NSSO is confirmed at this stage. The onboarding with the OAuth2 Authorization Server is managed by our technical teams, based on the information you sent.

6. Implementation and tests

Here is the main step, when you are going to implement your own MessageRegistry, based on the API and the development guidelines given in the next section. Of course it is important to test your service and the security before proceeding to the next step.

7. Integration of your WS & configuration (PRD)

At this step, you can deploy a test version of your MessageRegistry that can be integrated into our Acceptance environment. We can then check the integration with the user interface of the federated e-Box. As soon as the verifications are conclusive, we carry out the necessary configurations directly in Production.

8. Deploy in Production

You can now deploy your service in the production environment. It is recommended to proceed with the different sanity checks before ‘going live’.

9. Production-ready

Your integration with the e-Box system can finally ‘go live’. Congratulations, you are now actually part of the e-Box galaxy!

DocProvider technical integration

Implement its own MessageRegistry in the form of WS REST respecting the common API.

- Provide details of the implementation carried out using the Meta-API;
- Integrate with the Authorization Server (Social Security) to protect access to its WS.

Points of attention for DocProvider Integration

<table>
<thead>
<tr>
<th>Error-handling</th>
<th>When you’re implementing the RESTful API, please do not forget to handle the possible error cases (cf. error codes defined in the API). 403 or 404, when AccessToken is valid?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• 403-code MUST be used only for a call to a resourceId that well exists in the current ebox, but is not available for</td>
</tr>
</tbody>
</table>
| Free-search on all the metadata | Several query-params are designed in the RESTful API to allow researches among the messages in a current ebox (see operation GET ebox/messages). The “q” query-param SHOULD allow a free text-search among a maximum of the following properties of the messages AND their related referenceData:
- messageId;
- subject;
- messageType (Id; Name; Description);
- senderApplication (Id; Name; Description);
- senderOrganization(Id; Name; Description);
- businessData;
- paymentData. |
| Languages translations support | Each property corresponding to a name or a description is defined as a TranslatedString, with possible values in FR, NL, DE, EN. In your implementation, you MUST provide a value at least for the 3 official national languages (FR, NL, and DE). Even if one of these translations is not available in your own system, you have to provide the most appropriate value instead (for example, if you have only the description in FR and NL, it’s good practice to use FR-value for DE-one). |
| Logging & Audit trail | Each DocProvider is responsible for logging calls to his MessageRegistry: retrieval of the messages list, consult a specific message, get the content of a document, etc. For the moment there is no constraint in the API at this level. |
| Meta-API to describe your WS implementation details | In each new version of your WS, you SHOULD describe your implementation details using the MetaAPI operations. These operations indicate which query-params are supported or not, especially for sorting. |
| Multiple or single value(s) for a single query parameter | By default, a query-parameter is expected only once by request. The other cases are explicitly indicated in the API, via 'collectionFormat: multi'. Error management at this level: you have the choice to ignore a non-compliant parameter or |
| **Read-status management (1/2): principle** | As explained in the RESTful API, you **MUST** consider that a message is marked as read only if one of its main contents has been retrieved (one attachment with the `mainContent` property=true; or the body if its `mainContent` property=true). |
| **Read-status management (2/2): mainContent indicator** | Consequently to the point above, each message **MUST** have at least one content (attachment or body) considered as the main. |
| **ReferenceData management (1/3): principles** | The ebox/enterprise RESTful API proposes 3 types of referenceData: `messageType`; `senderOrganization`; and `senderApplication`.  
  - A `senderOrganizationId` **MUST** be the BCE/KBO number of the related organization.  
  - A `senderOrganizationShortName` and `LongName` **SHOULD** match the official names, except in case of valid business reason.  
  - `senderApplication` referenceData is not necessarily required. If provided, it is good practice to refer to data known to the end user (for example, online service names and descriptions).  
  - `MessageType` is required to categorize a message in the form of a well-known business category for several documents.  
  Each referenceDataId used as property of a message **MUST** correspond to a well-defined referenceData. |
| **ReferenceData management (2/3): cross-references** | In the details of a `referenceData`, it is planned to list the identifiers of the others referenceData related to the current one. For example, for a given `senderOrganization`, the lists of related `senderApplicationId(s)` and `messageTypeId(s)` are provided. |
| **ReferenceData management (3/3): logos (images) support** | A logo **MUST** be a PNG or GIF image on a transparent background. The three sizes proposed for the images **MUST** respect the following dimensions:  
  - Small <100 pixels (height = width);  
  - Medium: between 100px and 500px;  
  - Large >500px.  
  Each logo of a `senderOrganization` or `senderApplication` **SHOULD** match the official one, except in case of valid business reason. |
| **As precised in the API, you can provide a different logo for each supported language.** |
| **RegisteredMail property** | This property of a message MUST only be used in the case of a registered mail delivery from an **accredited service**. Any registered mail must be opposable, typically on a legal basis. Each DocProvider is responsible for the proper use of this flag for its messages. |
| **Security of your WS** | Each MessageRegistry must be protected based on the OAuth2 standard. It is planned to be integrated with the Social Security OAuth2 Authorization Server. Please see the “Annex 2: OAuth2 Authorization Server onboarding” for this point. |
| **Supporting direct-integration of your WS by a DocConsumer** | An enterprise who gets permission to become DocConsumer must be able to directly integrate your WS. |
| **Validity Period of the messages** | A message made available on your WS must be available during its validity period. This period is defined by message type, and must be indicated in the corresponding referenceData. Naturally, the expirationDate of a message depends on this period of validity. An expired message can never be returned by the WS. |
| **Verification of data-access** | As a new DocProvider, you are responsible to manage the data-access for all your messages, in order to allow only the authorized enterprises and users to access data they can consult. |
| **Versioning** | The version number of your WS is to be specified in the ProviderRegistry. In parallel, the version number of the API that your WS implements must also be specified. |
| **Visible status of the messages** | The “visible” property of a message must be implemented by all DocProviders, in order to allow the users of the e-Box interface to manage this status. It is therefore necessary to manage the change of visibility for a specific message, and for a list of messages (see PATCH operations in the API). |
Annexes

Annex 1: X.509 certificate

- Attention, you need a **distinct certificate for each work environment** (Acceptance, Production).
- The type of certificate requested must be a **non-public trust (applicative) certificate**.
  - Trusted CA: QuoVadis Trust Anchor Issuing CA G2;
- The order for a new certificate must be made via [info.be@quovadisglobal.com](mailto:info.be@quovadisglobal.com)
  - The mail must be sent in Dutch or English.
- The certificate format **MUST respect** the following structure:

<table>
<thead>
<tr>
<th>X509 QuoVadis</th>
<th>Value</th>
<th>Comment / example</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>BE</td>
<td>C = BE</td>
</tr>
<tr>
<td>O</td>
<td>ORGANIZATIONNAME</td>
<td>The legal name of your organization. Example: O = SMALS</td>
</tr>
<tr>
<td>OU</td>
<td>Belgian Federal Government</td>
<td>OU = Belgian Federal Government</td>
</tr>
<tr>
<td>OU</td>
<td>urn:be:fgov:kbo-bce:organization:cbe-number:xxxxxxxxxxx</td>
<td>Example for institution with CBE 01234567689 this OU must be</td>
</tr>
<tr>
<td></td>
<td>OU</td>
<td>OU = urn:be:fgov:kbo-bce:organization:cbe-number:0123456789</td>
</tr>
<tr>
<td>OU</td>
<td>ACC</td>
<td>Possible values {PRD, ACC, INT}. Example: OU = ACC</td>
</tr>
<tr>
<td>CN</td>
<td>ApplicationID-URN-from-request-document</td>
<td>Example for application werkkart/carte de travail (RVA/Onem), CN must be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN = employment:job-attest:werkkart</td>
</tr>
<tr>
<td>L</td>
<td>Brussels</td>
<td>The city where your organization is located. Example: L = Sint-Gillis</td>
</tr>
<tr>
<td>S</td>
<td>Brussel-Hoofdstad</td>
<td>The state/region where your organization is located. This shouldn't be abbreviated. Example: S = Brussel-Hoofdstad</td>
</tr>
</tbody>
</table>

- If necessary, the CSR can be sent to us for validation before the application at QuoVadis.

! Do not forget to have your “**e-Box DocProvider onboarding form**” validated (by [eBoxIntegration@smals.be](mailto:eBoxIntegration@smals.be)) before ordering your certificates to QuoVadis.
Annex 2: OAuth2 Authorization Server onboarding

This annex provides the necessary explanation to perform the OAuth 2.0 aspects of the integration as a DocProvider. This document will not cover the OAuth 2.0 protocol or the onboarding procedure. You can get more information directly from the spec or from some of the numerous internet resources available.

**Token validation & analysis**

As a DocProvider you will receive OAuth 2.0 tokens from clients and will have to determine:

- if that token is valid;
- who is the user attached to this token;
- what are the scopes that the user has.

All this will have to be done by using the introspect method of OAuth. To access the introspect method, you will have to request a specific token from the OAuth 2.0 server, the `scope:security:authorization:oauth:oauth-azt-server:introspect` scope. Use the "introspect token" to validate that your "client's token" is valid.

Note: The introspect method is available on the following URL's
ACC: [https://services-acpt.socialsecurity.be/REST/oauth/v3/introspect](https://services-acpt.socialsecurity.be/REST/oauth/v3/introspect)
PROD: [https://services.socialsecurity.be/REST/oauth/v3/introspect](https://services.socialsecurity.be/REST/oauth/v3/introspect)

Once you have a reply from the introspect endpoint, you will have to pay particular attention to the following fields of the response:

- **active**: Indicate that the token can be accepted by your server
- **principalAttributes**: contains information about the user attached to the token
  - `urn:be:fgov:kbo-bce:organization:cbe-number`: The CBE number of the company
  - `urn:be:fgov:person:ssin`: The SSIN attached to this token. This key is only available if the token is produced by a user via the Web App
- **exp**: The expiration date of the token expressed in seconds January 1 1970 UTC
- **scope**: The list of scopes associated with the token

**Reminder about identity**

Functionally speaking an eBox is identified by the CBE number of the company. However you are free to limit the visible items in the eBox based on other criteria.

When a user access the webservice via the Web App you can retrieve its SSIN (NRN = National Register Number) to do extra access control.

**Reminder resources access control**

Each scope gives access to a set of resources

  - `/ebox`
Example

An example project written in Java exists and illustrate how to validate an access token using the introspect resource. It also shows how to retrieve the most important keys from it.

This example uses nimbus, which we recommend for this task federeOauthTestApp.

Annex 3: References

G-Cloud REST guidelines: https://www.gcloud.belgium.be/rest/

OpenAPI v2 Specification (OAS) [aka Swagger] standard: https://swagger.io/specification/v2/


Website with technical information about e-Box Enterprise:
https://members.intranext.smals.be/projects/EBOX/default.aspx [ask access to eBoxIntegration@smals.be]