BOSA signature solution

|  |  |  |
| --- | --- | --- |
| ***Date*** | ***Author*** | ***Comment*** |
| *25.02.2021* | *Zetes* | *Initial version* |
| *12.03.2021* | *Zetes* | *Fixes in the JSON, updates in redirect and call back URLs* |
|  |  |  |

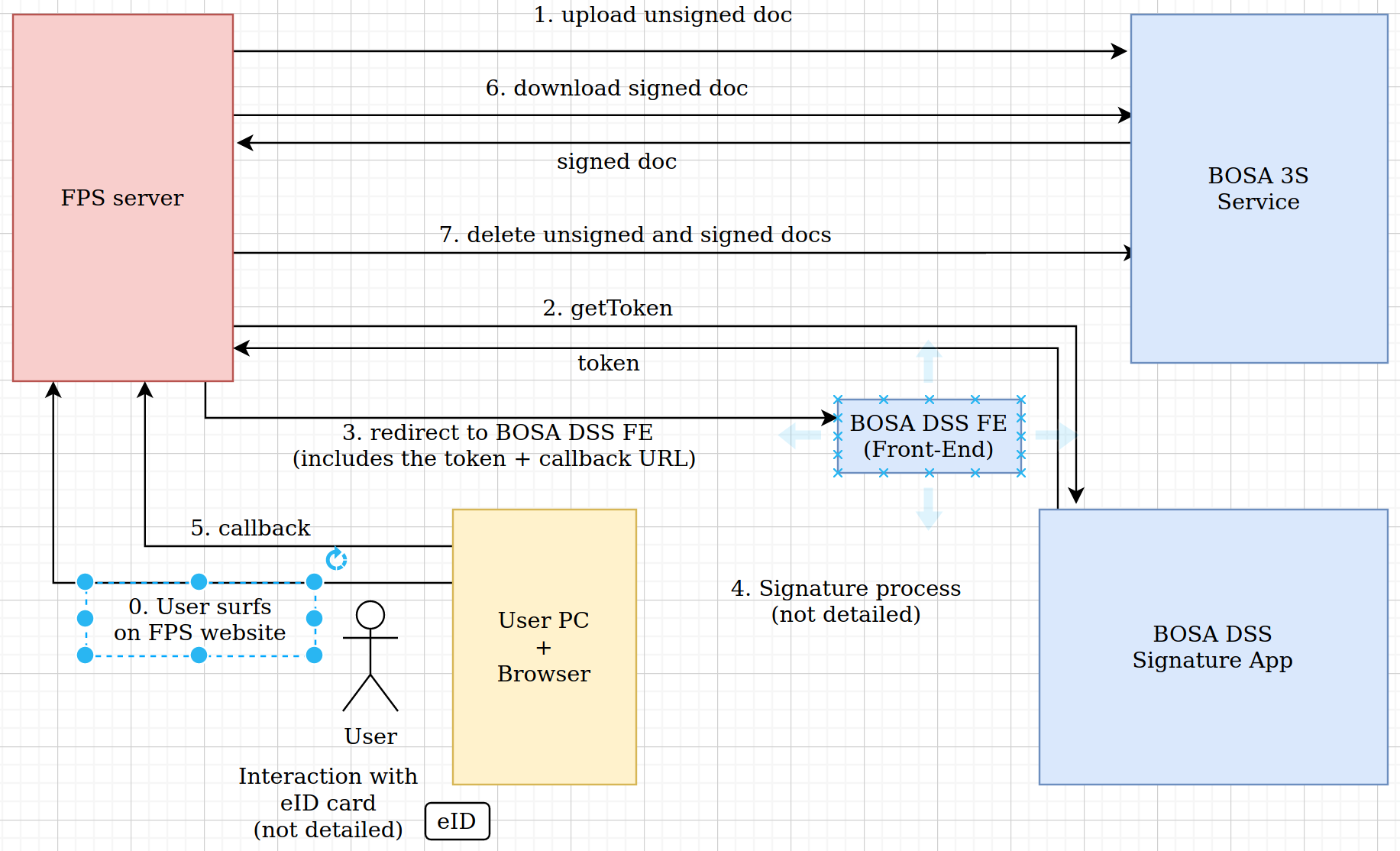
# 1. Introduction

The BOSA signature solution consists of 3 parties:

* An **FPS** that wants a User to sign a document with his/her eID/foreigner card
* A **User** with his/her PC, surfing on the FPS website
* **BOSA**, who offers the signing services

The flow goes as follows:

1. The User is surfing on the FPS website
2. The FPS creates a document (to be signed by the User) and uploads it to the BOSA S3 server
3. The FPS sends a ‘getToken’ request to the BOSA DSS server and gets back a *token*
4. The FPS redirects the User to the BOSA DSS front-end server, this HTTP redirect includes the token and a *callback URL*
5. The actual signature process is done between the User and BOSA, at the end a signed document is put on the BOSA S2 server
6. A callback to the FPS is done
7. The FPS downloads the signed document
8. The FPS deletes the unsigned and signed documents



# 2. Details

## 2.1. BOSA S3 server

BOSA provides an Amazon S3 compatible server: MinIO [1].

The FPS uses this server to:

* Upload unsigned documents
* Download signed documents
* Delete the unsigned and signed documents

When the FPS registers for the signature solution, it receives a username and password for this BOSA S3 server. With this, the FPS can then upload, download and delete documents.

*It is important to note that this BOSA S3 server is not intended for long time storage: the space allocated to an FPS is limited, and the FPS itself is responsible for timely deleting documents that are no longer needed.*

There are free client SDKs available for several languages (Java, python, .NET, …) on [2].  
An FPS can use these to implement the upload/download/delete calls.

Implementation note: to connect to an S3 server, typically 3 things are needed: a ‘bucket name’ (can be seen as a directory on the S3 server), a username and a password.  
For the BOSA S3 server however, the ‘bucket name’ and username are the same, therefore only a username and password are provided during the registration.

## 2.2. The getToken call

This is a REST call (a HTTP POST containing a json string) from the FPS to the BOSA DSS server; the BOSA DSS server will return a ‘token’.

This token is a string that contains the names of the unsigned (‘in’) and signed (‘out’) documents. It is cryptographically protected so that the Users can’t modify it.

For the FPS, this token is transparent and doesn’t have to be parsed/interpreted/.. It only has to be sent in the HTTP redirect to the BOSA DSS front-end server.

Here is an example of the contents of a getToken request:

|  |
| --- |
| *{*  *"name":"minfin",*  *"pwd":"U84SnLEQvp",*  *"in":"test.pdf",*  *"out":"signed\_test.pdf",*  *"prof":"XADES\_1"*  *}* |

The “prof” parameter specifies which type of signature is requested. BOSA will provide a list of the available values and their meaning.

## 2.3. The HTTP redirect

After the FPS has uploaded the document and obtained the token, it will do a HTTP redirect of the User’s browser to the BOSA DSS front-end server; this will handle the actual signing process.

This redirect contains the following parameters:

* The token obtained in the getToken call: this is at the end of the URL path
* A callback URL to the FPS: this is a query parameter
* A language parameter: this is a query parameter. Values: en, nl, fr, de

Example of a redirect URL:

|  |
| --- |
| *https://sign.ta.fts.bosa.belgium.be/sign/SDZEgF935d6e9.DKEi8D7nFDZz8EEZLazemJFeLIE.2FIDsdf987dILEJFd?callback=http%3A%2F%2F%2Feservices.minfin.fgov.be%2Fcallback%3Ffilename%3Dsigned\_test.pdf&language=en* |

## 2.4. The callback to the FPS

After the document is signed and available on the BOSA S3 server, the User’s browser will do a callback to the callback URL specified in the redirect (see above). The callback URL is completely specified by the FPS and can contain query parameters like a session ID.

The User’s browser will add 2 extra query parameters:

* err: an error code (0: success, 1: user cancelled, …)
* details: a string with error details, is absent if err = 0
* ref: a unique reference to the error, can be absent

Examples of a full callback URL:

|  |
| --- |
| ***http://eservices.minfin.fgov.be/callback?filename=signed\_test.pdf****&err=0* |

|  |
| --- |
| ***http://eservices.minfin.fgov.be/callback?filename=signed\_test.pdf****&err=1&details=user\_cancelled* |

|  |
| --- |
| ***http://eservices.minfin.fgov.be/callback?filename=signed\_test.pdf****&err=4&details=server\_problem&ref=20210219134552013* |

The bold part is the original callback URL specified by the FPS in the redirect, the rest has been added by the browser’s javascript.

# 3. Sample/test FPS service

A simple Java service has been made that can serve as an example for the FPS.

It provides a simple ‘home page’ where the use can select a document to sign.  
Then the sample service uploads this document, requests a token and does a redirect.  
Finally a callback page is foreseen where the signed document is downloaded and the unsigned and signed document are deleted from the S3 server.

To build the sample: install maven and type the following on a command prompt:  
 *mvn package*

To run the sample service on Linux/MacOSX:  
 *java -cp "target/lib/\*":target/test\_fps-0.0.1-SNAPSHOT.jar com.zetes.projects.bosa.testfps.Main*Or on Windows:  
 *java -cp "target/lib/\*";target/test\_fps-0.0.1-SNAPSHOT.jar com.zetes.projects.bosa.testfps.Main*

When started successfully, the service will show the URL to which to surf to:

|  |
| --- |
| *Service started - press Ctrl-C to stop*  *Surf with your browser to http://localhost:8080* |

When the service starts, it reads configuration parameters from the config.txt file.

# References

[1] MinIO: <https://en.wikipedia.org/wiki/MinIO>

[2] MinIO SDKs: <https://docs.minio.io/> - select ‘MINIO SDKS’ on the left