DG Digitale Transformatie
FOD Beleid en Ondersteuning
DG Transformation digitale
SPF Stratégie et Appui

TECHNICAL SPECIFICATIONS HANDBOOK RELATED TO THE ROYAL DECREE OF RECOGNITION OF PARTNER'S ELECTRONIC IDENTIFICATION SERVICES

FPS BOSA - DG Digital Transformation
Domain Identification, Authentication and Authorisation (IAA)

Table of Content

1 DOCUMENT CONTROL

BOSA-DT IAA Governance mandates this section in order to provide enough track records for this document.

1.1 DOCUMENT COPYRIGHT AND CLASSIFICATION

1.2 DOCUMENT HISTORY

Version	Date	Author	Description of changes/ Remarks
V.1.0	27/10/2017	BOSA-DG DT IAA	First published version
V.2.0	22/09/2020	BOSA-DG DT IAA	Second published version, updated with Universal Link method and signing of the authorization code request

1.3 CONTACT

For information on this document, contact

Service Desk BOSA DT:

ServiceDesk Digital Transformation <servicedesk.dto@bosa.fgov.be>
+32 (0)2 740 74 74

2 INTRODUCTION

2.1 SCOPE OF THIS DOCUMENT

This handbook defines the technical specifications an electronic identification service should comply with to be recognized as an electronic identification service for public applications according to the Royal Decree of 22 October 2017[1]. This handbook is intended to specify the technical interaction between the FAS and the Partner's identification and authentication infrastructure.

The following subjects are not covered and are out of the scope of this handbook:

- The details of the authentication method used by the partner;
- The registration process of the authentication means;
- The interaction between the eGov application and the FAS.

2.2 AUDIENCE

The target audience is the candidate partners for recognition of an electronic identification service.

2.3 IMPORTANCE OF THE REQUIREMENTS

In this document, the following wording is used to denote the importance of requirements:

The words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Term	Signification and relevance
MUST	This word, or the terms "**REQUIRED**" or "**SHALL**", mean that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase "**SHALL NOT**", mean that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective "**RECOMMENDED**", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

[1] Arrêté Royal du 22 octobre 2017 fixant les conditions, la procédure et les conséquences de l'agrément de services d'identification électronique pour applications publiques / Koninklijk besluit van 22 oktober 2017 tot vaststelling van de voorwaarden, de procedure en de gevolgen van de erkenning van diensten voor elektronische identificatie voor overheidstoepassingen

SHOULD NOT	This phrase, or the phrase "**NOT RECOMMENDED**" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the adjective "**OPTIONAL**", mean that an item is truly optional.

2.4 TERMINOLOGY

Basic terminology is defined in article 1 of the Royal Decree of 22 october 2017.

This specification is also based on terms defined by OAuth 2.0 and OpenID Connect 1.0 specifications.

The following complementary definitions are applicable.

Term	**Signification and relevance**
Application owners/Developers	The ones who develop and manage eGov public applications (web applications, mobile applications, etc …) that will use the FAS to identify and authenticate their end users.
Assurance level	Or “LOA” - As defined in the COMMISSION IMPLEMENTING REGULATION (EU) 2015/1502 of 8 September 2015
eGov application	Also called Public application or client application. Any application made available by a public service.
End user	Refers to “Utilisateur/Gebruiker” as defined in article 1 of the Royal Decree of 22 october 2017.
FAS	The Federal Authentication Service delivers a central standard Identification and authentication service as well as guidance/guidelines for developers, applications owners and Partners. The FAS is powered by FPS BOSA/DG DT/IAA.
In-band (IB) authentication scenario	The authentication process is executed on the same communication channel as the one used to access the protected resource.
Intent	Inter-process communication mechanism on mobile devices. (e.g. https://developer.android.com/reference/android/content/Intent.html)
Universal link	Universal linking is Apple's method of launching apps on iOS when linked from a website. Universal Links are a particular protocol for deep linking that are exclusive to Apple devices (available in iOS version 9 and above).
Native application	As defined in IETF Request for Comments 6749 Section 2.1 (RFC 6749).

Out-of-band (OOB) authentication scenario	The authentication process is executed on another communication channel than the one used to access the protected resource.
Partner	Refers to “Prestataire de services/Dienstverlener” as defined in article 1 of the Royal Decree of 22 october 2017.
Partner’s authenticator app	A native application delivered by the Partner in order to perform the cryptographic authentication action.
Push notification	A message from an external source that is sent to the mobile device and that can eventually trigger (depending on device) a specific application on the mobile device.
Unique Identification Number	Unique identifier of an end-user. It must be be the National register number (Numéro de registre national/Rijksregisternummer, also called NRN/RRN) or the Crossroads number (Numéro de banque carrefour / Kruispuntbanknummer, also called BIS number)
Web application	As defined in IETF Request for Comments 6749 Section 2.1 (REF 6749).

3 INTEGRATION SPECIFICATIONS

3.1 OVERALL CONTEXT FOR PARTNER AUTHENTICATION

3.1.1 Actors

Different target groups are involved into the model and have different functions and needs:

- **FAS:** It is the central authentication server and delivers guidance/guidelines for developers, applications owners and Partners. In the context of recognition of electronic identification services, FAS will delegate the identification and authentication phases to Partners
- **Partners:** They provide one or several recognized authentication service(s) with the best usability and security to the end users.
- **Application owners/Developers:** they develop and deploy egov web applications as well as egov native apps for Android and IOS. They interface with the FAS and not with the Partners directly.
- **End users**: They use their device to authenticate on an egov application running on any device.

3.1.2 Support of in-band and out-of-band authentications

The FAS will rely on Partners to provide identification and authentication of end users on eGov applications. The authentication means used in the Partner authentication scheme can run on any device. However, a big focus is put on authentication means with a front-end located on mobile devices. This is typically a native app on IOS and/or Android that will serve as authenticator.

The Partner MUST provide a complete solution delivered as a service covering the registration of the end user and the authentication of the end user.

The FAS will connect with the Partner's service via the federation protocol OpenID Connect. In the context of federation, the FAS MUST act as a client of the Partner and the Partner' service MUST NOT be aware for which eGov application it will act as an authenticator.

The Partner's service MUST support:

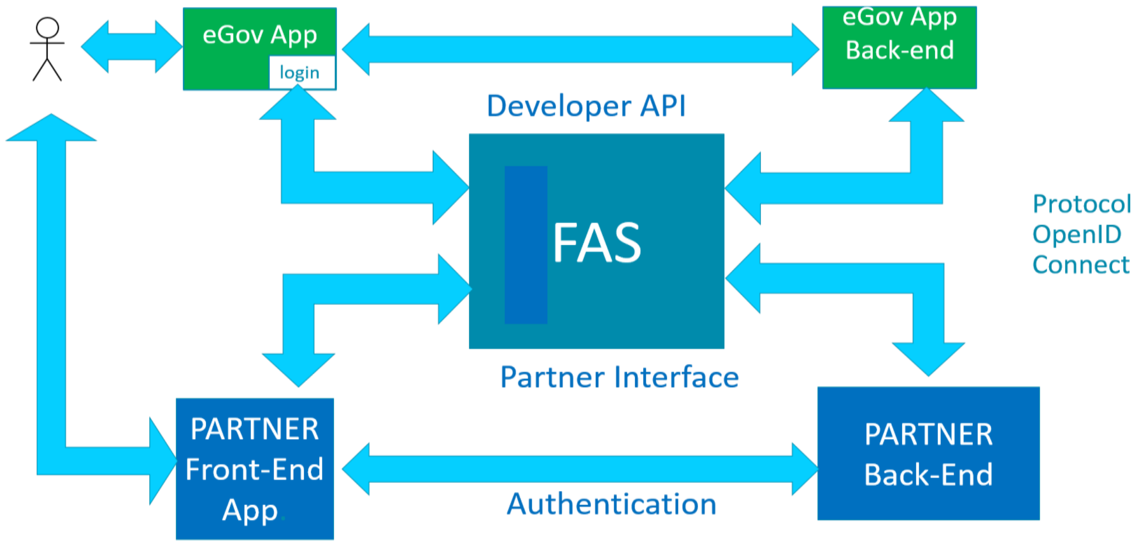
- End user authentication on web applications (browser based) running on any device.
- End user authentication on native and hybrid apps running on any device (Incl. PC/Mac and mobile devices).

If the authenticator app provided by the Partner runs on a mobile device (e.g. phone or tablet) device, then:

- It MUST be available for the Android AND iOS platform
- It MUST be available for downloads from the Apple App Store and Google Play Store and as such certified for compliance with the respective OS.

3.1.3 Federation Framework

The federation framework can be designed as follows :



The global FAS Partner authentication consists of multiple federation circles of trust (CoT):

- eGov circle of trust:
  - Includes the eGov applications, application owners and developers with the FAS serving as their IDP (Identity Provider).
  - The eGov CoT supports two federation protocols: SAML2.0 and OpenID Connect.
  - For native mobile apps, only the OpenID Connect protocol is supported and app developers MUST use this protocol.
  - Developers of eGov applications MUST use the session management and logout options of OpenID Connect
- Partners circle of trust:
  - Each Partner has his own circle of trust and a dedicated connection to the FAS, the FAS being their client. The partner considers the FAS as an IDP Proxy.
  - The Partner SHOULD NOT maintain a “session” on its infrastructure after end user authentication. If the Partner keeps a session after end user authentication, then this session MUST not last more than 120 seconds.
  - OpenID Connect ACR Values MUST be used to specify the minimal assurance level of the authentication means requested by the FAS.
  - The Subject Identifier(sub) in the ID Token provided by the Partner MAY contain the Unique identification number
  - The Partner MUST implement one of the following solutions:
    - The ID Token provided by the Partner will contain the Unique identification number of the authenticated person in a custom claim.
    - The Partner allows the FAS to retrieve the Unique identification number of the authenticated person by means of the userinfo endpoint.
  - It SHOULD NOT be possible to run the Partner authenticator app on a mobile device that is rooted or jailbroken. If the Partner authenticator app can run on such a device, then:
    - The end user MUST be informed of the risks and give his explicit consent;
    - The Partner will take measures to implement extra protection to the app and/or implement an anti-fraud functionality.

3.2 COMPONENTS VIEW AND INTERACTIONS

3.2.1 Description of the Log in procedure to an eGov application

The end user who wants to authenticate to an eGov application is always be redirected first to the FAS in order to choose a Partner for authentication.

In case of a native eGov app on mobile, the app will open the default browser of the mobile device and will send an OpenID Connect Authorization Request to the FAS where the end user will choose the Partner he wants to use to authenticate. Consequently, the interaction of the end user with the Partner authentication solution MUST always start from a browser.

When choosing the Partner “button”, an OpenID Connect authorization request is sent to the Partner’s OpenID Provider (OP in OpenID Connect). There are multiple scenarios. Here are some examples:

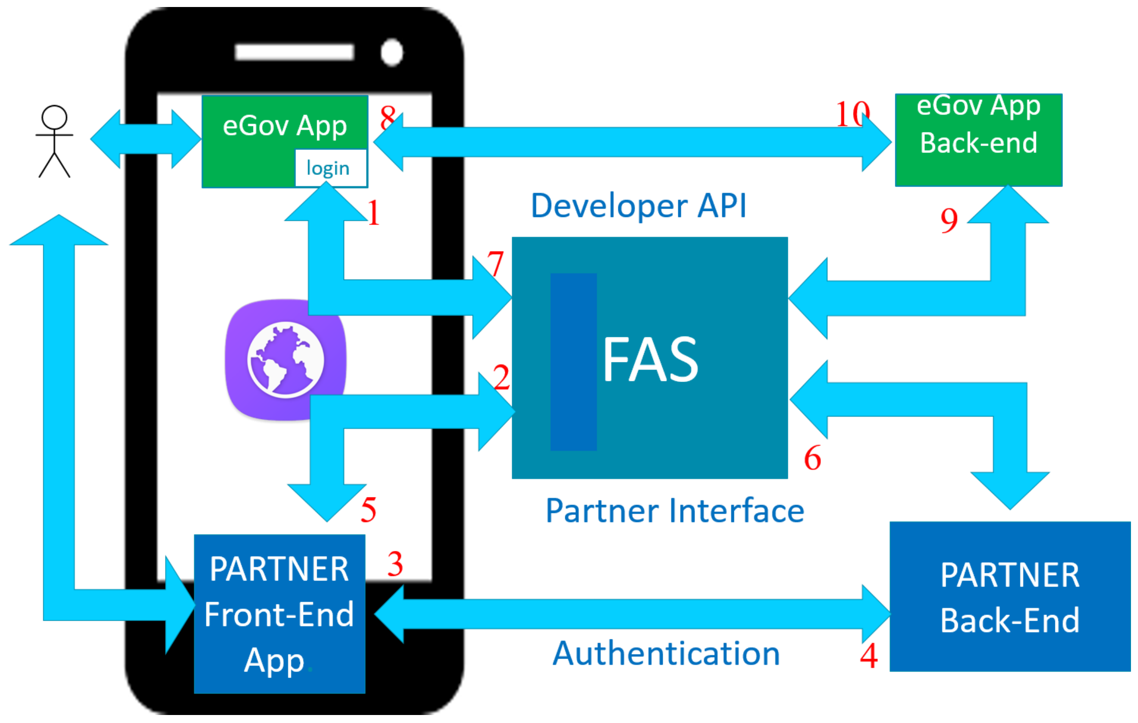
- Authorization request is sent to the authenticator front-end (the Partner’s authenticator native app) which pops-up on the device in case of an in-band scenario on mobile.
- Authorization request is sent to the authenticator back-end (the Partner authorization server) typically in an out-of-band scenario.

In case of mobile authentication, three main technologies MAY be supported to awaken the authenticator front-end: push notification,web-to-app intent or universal linking onto the mobile device.

Once authenticated at a Partner service, the authorization code from the Partner is transferred to the FAS that can retrieve an access token and user information from the Partner. Subsequently a session is created on the FAS.

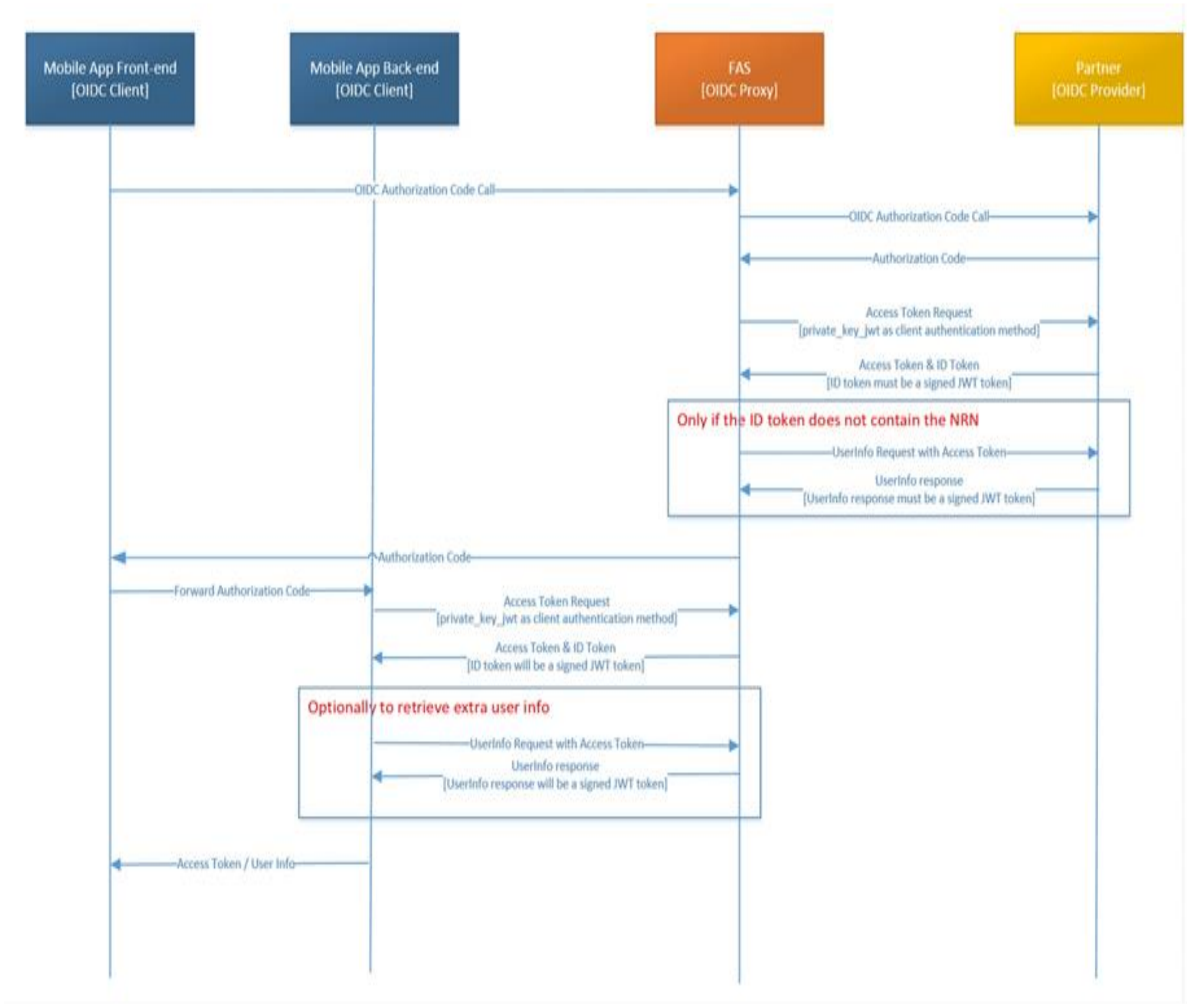
The FAS then sends its authorization code back to the client application. The back-end of the client app can then retrieve an access token and user information from the FAS. Subsequently the end user is fully authenticated.

Example interaction for an in-band scenario with a native mobile app:



1. End user clicks on the 'Log in with FAS' button in the native eGov app.
2. The native eGov app MUST open the default web browser on the mobile device and send an authorization code request to the FAS login page (Partner selection).
3. The end user selects the Partner and begins interaction with the Partner. An authorization code request is sent to the Partner
    a. Intent or universal link on the mobile device: the Partner button triggers an intent or universal link (web-to-app) to the Partner app on the mobile device and the end user authenticates via the app.
    b. Push notification: the Partner button redirects the end user to the Partner's website where the end user enters his identification credentials. A push notification is sent to the pre-registered device and triggers the mobile authenticator app where the end user authenticates.
4. The end user authenticates via the app and an authorization code is sent back to the FAS via the browser on mobile.
5. With the Partner's authorization code the FAS retrieves an access token and ID Token from the Partner. Optionally the FAS can retrieve extra user information from the Partner's user-info endpoint.
6. The FAS then sends back an authorization code to the eGov app via the browser (web-to-app).
7. The native eGov app on mobile communicates the authorization code to the eGov app's back-end.
8. With the FAS authorization code the eGov app's back-end retrieves an access token and ID Token from the FAS. Optionally, the eGov app's back-end can retrieve extra user information from the FAS user-info endpoint using the access token retrieved from the FAS.
9. The eGov app's back-end communicates the access token and user information to the eGov app on mobile.
10. The user is now identified and authenticated in the native eGov app.

Sequence diagram for the OpenID Connect flows:



Mobile App Front-end [OIDC Client]
Mobile App Back-end [OIDC Client]
FAS [OIDC Proxy]
Partner [OIDC Provider]
OIDC Authorization Code Call
OIDC Authorization Code Call
Authorization Code
Access Token Request [private_key_jwt as client authentication method]
Access Token & ID Token [ID token must be a signed JWT token]
Only if the ID token does not contain the NRN
UserInfo Request with Access Token
UserInfo response [UserInfo response must be a signed JWT token]
Authorization Code
Forward Authorization Code
Access Token Request [private_key_jwt as client authentication method]
Access Token & ID Token [ID token will be a signed JWT token]
Optionally to retrieve extra user info
UserInfo Request with Access Token
UserInfo response [UserInfo response will be a signed JWT token]
Access Token / User Info

3.3 PARTNER INTERFACE SPECIFICATIONS

The FAS acts as an OpenID Connect client and the Partner as an OpenID Connect provider.

Partner MUST only support the authorization code flow. Partner MUST NOT support the implicit or hybrid authorization flows.
With the OpenID Connect authorization code flow, the ID token and access token MUST be retrieved from the token endpoint. And more user information MAY be retrieved from the Partner's user info endpoint.
The Partner MUST expose a token endpoint and MAY expose a user info endpoint.
Private_key_jwt must be used as client authentication mechanism.

3.3.1 OpenID Connect authorization code request and response

The authorization code request starting from the FAS will be signed. Additionally, the "state" parameter MUST be used to avoid cross-site request forgery attacks. The state parameter is a non-guessable string known only to the Partner application, the Partner MUST make sure that the value has not changed between requests and responses. This is clearly stated in the "Security considerations" section 10.12 "Cross-Site Request Forgery" of RFC 6749 (OAuth 2.0) and in section 3.5 of RFC 6819.

GET towards the authorization code endpoint of the Partner:

Parameter		
scope	MUST contain	Openid and the (to be agreed) scope value to request the claim containing the unique identification number
response_type	MUST be	Code
client_id	MUST be	Value to be agreed
redirect_uri	MUST include	The https scheme
state	MUST contain	An opaque value used to maintain state between the request and the callback
response_mode	MUST NOT be used	
nonce	MUST NOT be used	A nonce is only used in implicit flows. FAS only supports the authorization code flow.
display	MUST NOT be used	
prompt	MUST NOT be used	
max_age	MUST NOT be used	
ui_locales	MUST be used	The partner must support all official Belgian languages (Dutch, French, German) and English
id_token_hint	MUST NOT be used	
acr_values	MUST be used	ACR values will be used to communicate the requested assurance level to the Partner
claims	MUST NOT be used	
request	MUST contain	Signed JWT containing AuthZ Code request claims

Request JWT **claims:**		
iss	MUST contain	client_id
aud	MUST contain	Authorization endpoint of the Partner

scope	MUST contain	Openid and the (to be agreed) scope value to request the claim containing the unique identification number
acr_values	MUST contain	ACR values will be used to communicate the requested assurance level to the Partner
redirect_uri	MUST	The https scheme
client_id	MUST contain	client_id
response_type	MIUST contain	code
state	MUST contain	An opaque value used to maintain state between the request and the callback

This request parameter containing a JWT MUST be digitally signed.

The FAS will sign the request JWT using RS256, RSA using SHA-256.

ACR values MUST comply with the eIDAS Specification and have one of the following values:

- http://eidas.europa.eu/LoA/high
- http://eidas.europa.eu/LoA/substantial

Example request:
GET to Partner authorization code endpoint
?response_type=code
&client_id=myClientID
&scope=openid%20profile
&acr_values= http://eidas.europa.eu/LoA/substantial
&redirect_uri=https://fedict.be
&state=af0ifjsldkj

Response Partner:
GET to the redirect uri of the authorization code request with the following parameters:

Parameter		
scope	SHOULD contain	The requested scopes
code	MUST contain	The authorization code
state	MUST contain	Must be the same value as in the authorization code request

Example response:
redirect_uri
?scope=profile%20openid
&code=f6b6181f-32bb-47ef-a825-34ad798c2938
&state=af0ifjsldkj
Sans cadre

The Partner will list all the possible token error responses; the responses must contain the 'error' (pre-defined error codes – OpenID Connect RFC) and the 'state' parameter (same value as in the authorization code request).

3.3.2 OpenID Connect Token Request

3.3.2.1 Client authentication

The OpenID Connect RFC describes 4 possible client authentication methods used to authenticate to the authorization server when using the token endpoint:

- Client_secret_basic
- Client_secret_post
- Client_secret_jwt
- Private_key_jwt

The FAS implementation only supports Private_key_jwt.

3.3.2.2 Request with Client Private_Key_JWT authentication

POST to the token endpoint of the Partner:

Parameters		
grant_type	MUST contain	The value must be ‘authorization_code’
code	MUST contain	The authorization code received from the Partner.
redirect_uri	MUST contain	Same redirect url as in the authorization code request
client_assertion_type	MUST contain	urn:ietf:params:oauth:client-assertion-type:jwt-bearer
client_assertion	MUST contain	A signed JWT token with the required claims in the following table

private_key_jwt claims		
iss	MUST contain	JWT issuer (Value to be agreed)
sub	MUST contain	clientID
aud	MUST contain	Token endpoint of the Partner
exp	MUST contain	An expiration time for the JWT
jti	MUST contain	A unique identifier for the token, to prevent the reuse of the token

This JWT token MUST also be digitally signed.

The FAS will sign the private_key_jwt using RS256, RSA using SHA-256.

The Partner can find the public key used for signature verification at jwks URI (to be agreed). The alg Header Parameter value of the JOSE Header will be set to an appropriate algorithm as defined in JSON Web Algorithms [JWA]. A kid value (Key ID) will also be provided in the JOSE Header in order to select the correct public key for signature verification.

In a later phase, optionally the following ECDSA asymmetric signatures will be supported by the FAS to sign the private_key_jwt:
• ES256 - ECDSA with SHA-256 and NIST standard P-256 elliptic curve
• ES384 - ECDSA with SHA-384 and NIST standard P-384 elliptic curve
• ES512 - ECDSA with SHA-512 and NIST standard P-521 elliptic curve

3.3.3 OpenID Connect Access/ID token response and unique identification number collection

Two scenarios will be supported:

1. The Unique identification number will be delivered via the ID token, using a specific scope/claim that will be agreed (syntax) by the FAS and the Partner.
2. The Unique identification number will be available after authentication at the userinfo endpoint of the Partner.

Access token response:
After receiving and validating a valid and authorized Token request from the client, the authorization server returns a successful response that includes an ID token and an access token. The response uses the application/json media type.

Parameters		
access_token	MUST contain	A valid access token
token_type	MUST be	Bearer
expires_in	MUST be	As low as possible, the value must not exceed 120 seconds.
id_token	MUST contain	An ID token value associated with the authenticated session and MUST be digitally signed
refresh_token	MUST NOT be used	

Example response access token request:

```
{
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "expires_in": 120,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
    NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3F0MyIsCiAibm9uY2UiOiAibi0wUzZ
    fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzR0EHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSP0aAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
 }
```

3.3.3.1 Scenario 1 - Unique identification number in the ID token

When choosing the ID token approach to send the unique identification number, the partner must send a signed JWT acting as ID token.
id_token (signed JWT) requirements:

Parameters		
iss	MUST contain	The issuer identifier for the issuer of the response, this URL must use the 'https' scheme
sub	MUST contain	A subject identifier
aud	MUST contain	the ClientID
exp	MUST contain	The expiration time should not be more than a few minutes
iat	MUST contain	The time at which the JWT was issued
auth_time	MUST NOT be used	
nonce	MUST NOT be used	
acr	MUST be used	Specifies the authentication assurance level the end user is logged in with.
amr	MUST NOT be used	
azp	MUST NOT be used	
claim	MUST BE used	This claim MUST contain the Unique identification number

The syntax of the claim can be defined by the Partner.

ID token MUST be digitally signed.

The FAS SHALL support by default the following RSA signatures:

- RS256 - RSA using SHA-256

AND in a later phase, optionally the following ECDSA asymmetric signatures will be supported by the FAS:

- ES256 - ECDSA with SHA-256 and NIST standard P-256 elliptic curve
- ES384 - ECDSA with SHA-384 and NIST standard P-384 elliptic curve
- ES512 - ECDSA with SHA-512 and NIST standard P-521 elliptic curve

When using RSA or ECDSA Signatures, the alg Header Parameter value of the JOSE Header MUST be set to an appropriate algorithm as defined in JSON Web Algorithms [JWA]. The private key used to sign the content MUST be associated with a public key used for signature verification published by the sender in its JWK Set document. If there are multiple keys in the referenced JWK Set document, a kid value MUST be provided in the JOSE Header. The key usage of the respective keys MUST support signing.

Example of decoded id_token:

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "acr": http://eidas.europa.eu/LoA/substantial  }
!
```

The Partner will list all the possible token error responses, the responses MUST contain the 'error' (pre-defined error codes – OpenID Connect RFC).

3.3.3.2 Scenario 2 - Unique identification number at user info endpoint

When choosing the userinfo approach to send the unique identification number, the partner MUST send a signed JWT.
Userinfo JWT requirements:

Parameters		
iss	MUST contain	The issuer identifier for the issuer of the response, this URL must use the 'https' scheme
sub	MUST contain	A subject identifier defined by the Partner
aud	MUST contain	the ClientID
exp	MUST contain	The expiration time should not be more than a few minutes
iat	MUST contain	The time at which the JWT was issued
auth_time	MUST NOT be used	
nonce	MUST NOT be used	
acr	MUST be used	Specifies the authentication assurance level the end user is logged in with.
amr	MUST NOT be used	
azp	MUST NOT be used	

Userinfo JWT MUST be digitally signed.

The FAS SHALL support by default the following RSA signatures:

- RS256 - RSA using SHA-256

AND in a later phase, optionally the following ECDSA asymmetric signatures will be supported by the FAS:

- ES256 - ECDSA with SHA-256 and NIST standard P-256 elliptic curve
- ES384 - ECDSA with SHA-384 and NIST standard P-384 elliptic curve
- ES512 - ECDSA with SHA-512 and NIST standard P-521 elliptic curve

When using RSA or ECDSA Signatures, the alg Header Parameter value of the JOSE Header MUST be set to an appropriate algorithm as defined in JSON Web Algorithms [JWA]. The private key used to sign the content MUST be associated with a public key used for signature verification published by the sender in its JWK Set document. If there are multiple keys in the referenced JWK Set document, a kid value MUST be provided in the JOSE Header. The key usage of the respective keys MUST support signing.

Example of the decoded userinfo JWT:
```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "acr": http://eidas.europa.eu/LoA/substantial
 }
```

The Partner will list all the possible token error responses, this response must contain the 'error' (pre-defined error codes – OpenID Connect RFC.

Collection of the Unique identification number at the user info endpoint:
The Unique identification number will be collected after authentication at the user info endpoint of the Partner. The Partner must return a signed JWT token that includes a claim (syntax will be agreed by the FAS and the Partner), which contains the Unique identification number.

This JWT token containing the RRN claim MUST be signed.

The FAS SHALL support the following RSA signatures:

- RS256 - RSA using SHA-256

AND in a later phase, optionally the following ECDSA asymmetric signatures will be supported by the FAS:

- ES256 - ECDSA with SHA-256 and NIST standard P-256 elliptic curve
- ES384 - ECDSA with SHA-384 and NIST standard P-384 elliptic curve
- ES512 - ECDSA with SHA-512 and NIST standard P-521 elliptic curve

When using RSA or ECDSA Signatures, the alg Header Parameter value of the JOSE Header MUST be set to an appropriate algorithm as defined in JSON Web Algorithms [JWA]. The private key used to sign the content MUST be associated with a public key used for signature verification published by the sender in its JWK Set document. If there are multiple keys in the referenced JWK Set document, a kid value MUST be provided in the JOSE Header. The key usage of the respective keys MUST support signing.

3.4 OTHER (NON FUNCTIONAL) SPECIFICATIONS

3.4.1 Security

These security requirements are inspired from the eIDAS technical specifications v1.2 especially the eIDAS - cryptographic requirements for the Interoperability Framework.

The Partner MUST show, annually, evidence that its solution is compliant to security standards.

TLS

- All communications MUST be encrypted using the TLS protocol with a version of the following list.
  - If supported by the citizen's browser, the connection MUST use TLS 1.2.
  - Prior TLS versions MUST NOT be used.
- All communications MUST only use cipher suites that provide perfect forward secrecy.
- The certificates MUST have sufficient key strength and a lifetime of maximum 5 years.
- There MUST be a certificate revocation list (CRL) setup by the Partner for the certificates being used by the Partner for the TLS handshakes.
- There SHOULD be an online verification service for the certificates (OCSP).
- For the TLS handshake, elliptic curves cryptography MUST be used with Diffie-Hellman key exchange (ECDHE-ECDSA, ECDHE-RSA, DHE-RSA …).
- The signature algorithm MUST be based on SHA-256 at a minimum.

PARTNER'S NATIVE APP ON THE MOBILE DEVICE

- At download time from the App Store and/or Play Store the app MUST NOT contain any long-term private key.
- If the installation process requires a login name or any third party password, it MUST be communicated in a secure TLS tunnel (see requirements above).
- The application SHOULD use certificate pinning to ensure there are no man-in-the-middle attempts to decrypt, modify and then re-encrypt any communications. It SHOULD check the public key fingerprint of the certificate (not the issuer). Public key SHOULD be stored in a configuration file with strict access permissions.
- The app MUST only request the necessary permissions: it SHOULD NOT request access to location, photos, Bluetooth, Wi-Fi state, audio, wallpaper settings, alarms or time zone settings unless the Partner is able to demonstrate the necessity.
- If the mobile application is built on a certain framework such as Cordova/Phonegap or Mono for Android, the Partner MUST monitor potential vulnerabilities in those platforms and MUST update the application accordingly.
- Sensitive data such as session keys, usernames or token values MUST BE stored encrypted. If a PIN is used, the PIN SHOULD only work to decrypt a key stored on the phone, which is then used to authenticate the application to the server. The PIN SHOULD never be sent to the server.
- If the application writes a log file, sensitive data MUST NOT be included in those log files.
- The application MUST be bound to the device. The application SHOULD NOT work when moved to another device.
- When using cryptographic technologies (x.509 certificates and keys etc …), the Partner MUST make sure algorithms/ciphers, keylength, timelife of keys/certificates will be managed and enhanced following the industry standards ETSI, NIST, PCI-DSS.

PARTNER BACK-END

- Audit logs MUST be protected for confidentiality and integrity logs and archives.
- End user information (user profile) MUST be protected for confidentiality and integrity.

3.4.2 Privacy

At registration time, the Partner MUST ensure that the end user is clearly informed about the transfer to and the use by the Partner and by FOD BOSA DG DT of his/her National register number or Crossroads number for the purpose of unique identification and authentication.

3.4.3 Usability

The solution MUST present a high degree of user-friendliness and intuitiveness and will as much as possible be accessible to users with disabilities.

When an end user wants to login using a recognised electronic identification service, only the (or the several) recognised services of the Partner must be shown and can be used for accessing an eGov application, according to the assurance level required.

3.5 OPERATIONAL SPECIFICATIONS

3.5.1 Integration Environment

An integration environment is used for integration and pre-production testing. FAS production and integration environments are completely separated, functionally and technically.

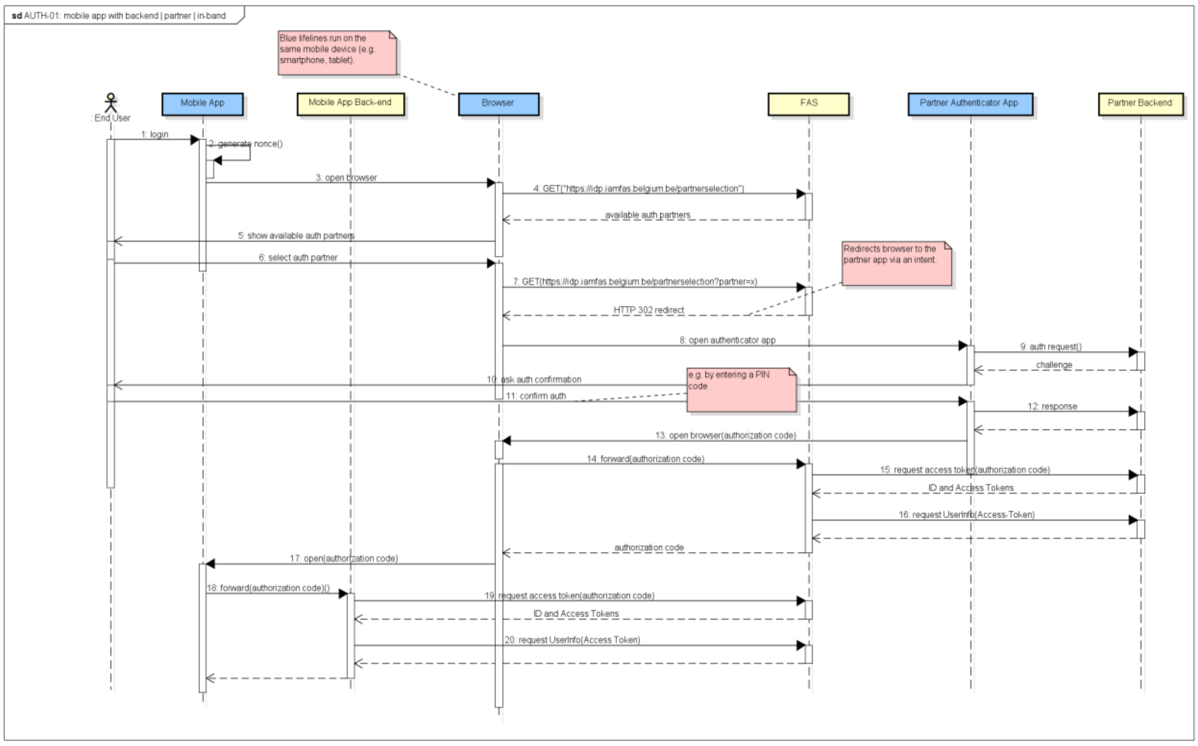
For the Partner integration and pre-production testing process, an integration environment at Partner side MUST be available and it MUST be separated from the production environment.

3.6 EXAMPLES OF USE CASES FOR AUTHENTICATION

3.6.1 In-band: native eGov app on mobile

Example 1: web-to-app intent on mobile (or universal linking)

- End user clicks onto the 'Log in with FAS' button in the native eGov app.
- The native eGov app opens the default web browser on the mobile device and navigates to the FAS login page (Partner selection).
- The user selects *Partner*.
- The Partner button triggers **an intent** (web-to-app) to the Partner app on the mobile device (if the app is available on the device).
- The end user authenticates by identifying himself on the authenticator app (entering his/her app PIN code or fingerprint or whatever the Partner supports).
- The Partner system verifies the identification of the end user and upon success considers the end user as successfully authenticated.
- Partner sends the end user back to the FAS.
- The FAS authenticates the end user who is redirected (intent) to the original native eGov app.
- The user is now authenticated in the native eGov app.


sd AUTH-01: mobile app with backend | partner | in-band
Blue lifelines run on the same mobile device (e.g. smartphone, tablet).
: End User
Mobile App
Mobile App Back-end
Browser
FAS
Partner Authenticator App
Partner Backend
1: login
2: generate nonce()
3: open browser
4: GET("https://idp.iamfas.belgium.be/partnerselection")
available auth partners
5: show available auth partners
6: select auth partner
7: GET(https://idp.iamfas.belgium.be/partnerselection?partner=x)
Redirects browser to the partner app via an intent.
HTTP 302 redirect
8: open authenticator app
9: auth request()
challenge
10: ask auth confirmation
e.g. by entering a PIN code
11: confirm auth
12: response
13: open browser(authorization code)
14: forward(authorization code)
15: request access token(authorization code)
ID and Access Tokens
16: request UserInfo(Access-Token)
authorization code
17: open(authorization code)
18: forward(authorization code)()
19: request access token(authorization code)
ID and Access Tokens
20: request UserInfo(Access Token)

Messages Details:

Authorization code request:
GET to [Partner 1 Authorization Endpoint URI]
?response_type=code
&client_id=etzujf89xxsd9ihq0iuhod7j9
&redirect_uri=https%3A%2F%2F[FAS Client Redirect URI]%2Ffas%2Foauth2c%2FOAuthProxy.jsp
&state=3625235475896327207589632720
&scope=openid%20profile%20email%20eid_nin
&acr_values=[see below]

Access token request (private_key_jwt as client authentication):
POST to [Partner Token Endpoint URI]

Parameters:
code=60PfJyF3yMZm0E8CzE9mRitob
grant_type=authorization_code redirect_uri=https%3A%2F%[FAS Client Redirect URI]%2Ffas%2Foauth2c%2FOAuthProxy.jsp
signed private_key_jwt token [see below]

Example Response:
{"uuid":"6cd6b890-559c-0134-bc0a-00163e43cad9","access_token":"xgAGyF4PLjqMLncPVKcOezQNEj0v5sv gQHvlYDSh","scope":"openid email profile eid_nin",”ID_token”:[ID_token]}

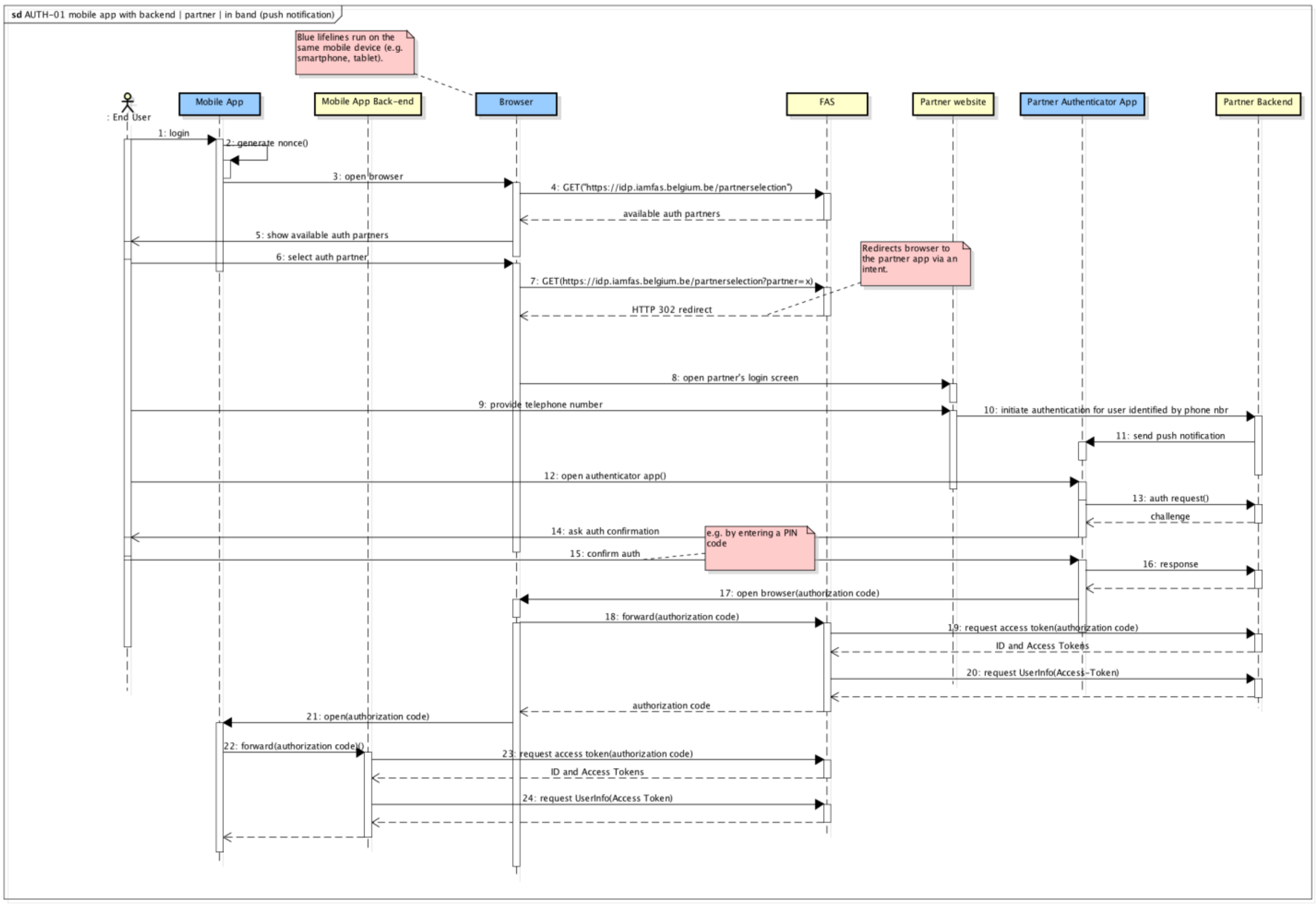
User info Request:
POST to Partner userinfo endpoint url

Header:
Authorization: Bearer 'Access-token-you-received-in-previous-call'
Example Response:
{"national_identification_number":"XXXXXXXXXX"}

The Partner can choose the naming of the unique identification number claim
Also the response from the userinfo endpoint should be a signed JWT, NOT plain JSON.

Example 2: push notification

- End user clicks onto the 'Log in with FAS' button in the native eGov app.
- The native eGov app opens the default web browser on the mobile device and navigates to the FAS login page (Partner selection).
- The end user selects *Partner*.
- The FAS redirects the end user to a Partner webpage.
- The Partner's site asks the end user to fill in his/her identification/telephone number to receive a **push notification.**
- The end user opens the push notification on his smartphone.
- The end user authenticates, e.g. by entering PIN or using fingerprint on the mobile app.
- The Partner system verifies the provided credentials and upon success considers the end user as successfully authenticated.
- Partner sends the end user back to the FAS.
- The FAS authenticates the end user and sends an intent (or uses universal linking) to the original native eGov app.
- The end user is now authenticated in the native eGov app.


sd AUTH-01 mobile app with backend | partner | in band (push notification)
Blue lifelines run on the same mobile device (e.g. smartphone, tablet).
: End User
Mobile App
Mobile App Back-end
Browser
FAS
Partner website
Partner Authenticator App
Partner Backend
1: login
2: generate nonce()
3: open browser
4: GET("https://idp.iamfas.belgium.be/partnerselection")
available auth partners
5: show available auth partners
6: select auth partner
Redirects browser to the partner app via an intent.
7: GET(https://idp.iamfas.belgium.be/partnerselection?partner=x)
HTTP 302 redirect
8: open partner's login screen
9: provide telephone number
10: initiate authentication for user identified by phone nbr
11: send push notification
12: open authenticator app()
13: auth request()
challenge
14: ask auth confirmation
e.g. by entering a PIN code
15: confirm auth
16: response
17: open browser(authorization code)
18: forward(authorization code)
19: request access token(authorization code)
ID and Access Tokens
20: request UserInfo(Access-Token)
authorization code
21: open(authorization code)
22: forward(authorization code)()
23: request access token(authorization code)
ID and Access Tokens
24: request UserInfo(Access Token)

Messages Details:

Authorization code request:
GET to [Partner Authorization Endpoint URI]
?client_id=FAS
&scope=openid%20service%3AFAS_Web_Login
&redirect_uri=https%3A%2F%[FAS Client Redirect URI]%2Ffas%2Foauth2c%2FOAuthProxy.jsp
&response_type=code
&state=omcjd6bjbjchtt5a2rctak5xtlwidre
&Acr_values=[see below]

Access token request ('private_key_jwt' as client authentication):
POST to [Partner Token Endpoint URI]

Parameters:
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer
code='the-authorization-code-you-received-in-previous-request'
grant_type=authorization_code
redirect_uri=https%3A%2F%2F[FAS Client Redirect URI]%2Ffas%2Foauth2c%2FOAuthProxy.jsp
client_assertion=(JWT token we produced) example:
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IjdPcjVGYkxheFJQeHRYYWZSUXhuK283R0s5cz0ifQ.ey
Jpc3MiOiJGRURJQ1QiLCJ
hdWQiOiJodHRwczovL3VhdG1lcmNoYW50LnNpeGRvdHMuYmUvb2lkYy90b2tlbiIsInN1YiI6IkZFRElDVCIsI
mV4cCI6MTQ3OTExODAw
MiwianRpIjoiaVVwRUFKdGx4ejh3VkhqSTIySXNPZyJ9.MuMeUORNbIy2xCZ8WUiyXlxNihITyPFKWC1H1KY
qW4X-IVWbR0grO4bE-TO
h4m_jnqHdzqP38gaseA9wE7iFyXRHqlcVW4qagQCnHT71xpCqpcZ12cdWeLTvPuxheVvA6DKNhfnd1x1M
DqnLYH4r1-7QrZH7h9ESTSj
yqdhWR7DqM498enQatrl5oK5nxqIkOrb5vWRx-2N3rzRYnH8bnech9yzIwOpO7jOtvgh3NAtHsLDqFz4b-
zsTCJfv5I0ITYswYSwaX2b-7F5
ebuaKaDUdi1fgLfEhPhXShFtTv6Y7AqOTtz4PISXIeVxIdNXaEad8qJv9jgTQb_h-h_KSrA

Example response:
{"access_token":"VBjcIa48tU0yHW8EvXGhgKNO5mZw-
Y0qerhFWZAqTWE","token_type":"Bearer","expire_in":300,"id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6I
nMxIn0.eyJleHAiOjE0NzkxMTgxODMsInN1YiI6ImxmOGZicjQ3bmphY3VjbWlpcXd
xbzhsajlxc3hoY3pqZ3Y5MiIsImF1ZCI6IkZFRElDVCIsImlzcyI6Imh0dHBzOlwvXC91YXRtZXJjaGFudC5zaXh
kb3RzLmJlX
C9vaWRjIiwiaWF0IjoxNDc5MTE3ODgzLCJhY3IiOiJ0YWc6c2l4ZG90cy5iZSwyMDE2LTA2OmFjcl9iYXNpYy
J9.TAu4KbR
coaIwRaWwNPKz2Autnn61JhuvJeKqU4Oo6AVPmqY4axg0nz1d1earMcD7iSVQpcsiEwrwWuolQd1FGdxuv
tmjfvYb7giVT
fUKDkIrxZJSIHflbxHTtIIquyjunroHCA1ESmClus-
iICDtsejoSz9MdwcFCNSl8dx0gXqlYrcyNQb9civSve3rUJE4s_8-L0PeoE
P79sfa6gC1v-kEk4kPr1vqCJMttVjUk-rxz0f68EvOBmjF5-
cbc1JT0OGAV1jXmauwAJcxpaaiDoZsj6Q70smIyg7gK0Ik5xGqA
Pvywpy6rSwpQe03MHSJVVLzPKOF7e_Ax0aaUmIqmQ"}

User info Request:
POST to [Partner User info Endpoint URI]

Header:
Authorization: Bearer 'Access-token-you-received-in-previous-call'

Example response:
eyJhbGciOiJSUzI1NiIsImtpZCI6InMxIn0.eyJzdWIiOiJsZjhmYnI0N25qYWN1Y21paXF3cW84bGo5cXN4aG
N6amd2OTIiL
CJpc3MiOiJodHRwczpcL1wvdWF0bWVyY2hhbnQuc2l4ZG90cy5iZVwvb2lkYyIsImF1ZCI6IkZFRElDVCJ9.v
zXQmga8TvS
W4WH0-
0EnJV76dL8hSXGdYlAC3y_sOh6Zd2hNYcLMpLHxLPJbPTJjPoacO3xZnCdBiz2XltDgoYPd76udjPPgtoW7_
wy0
GRX8RqmbBlpLkk5U8KVXXLCNOoVy1CxGZezh5gAgqVdFXRoM_M7AejZ8TsjqImWtcywuITfLfWhNmfWT
m1K19oEkn3
YNGlGhKYehIq3HeZ4H9ycrjXwMzI1uBAC0oU8yALVAPKiB1fNHiI9_rWS5rbbS0ljOZp4bKLvGs81g1tD9JqSg
-xNPNmb0K
IpQjHpyxp-pyud6A9oUlJ6sAJOYUklD3G41KVMF9Ts_I7LF4H0vSQ

BASE64 decoded & signed by Partner:
```
{
"alg": "RS256",
"kid": "s1"
}
{
"sub": "lf8fbr47njacucmiiqwqo8lj9qsxhczjgv92",
"iss": "https:// Partner2.be/oidc",
"aud": "FAS"
}
```

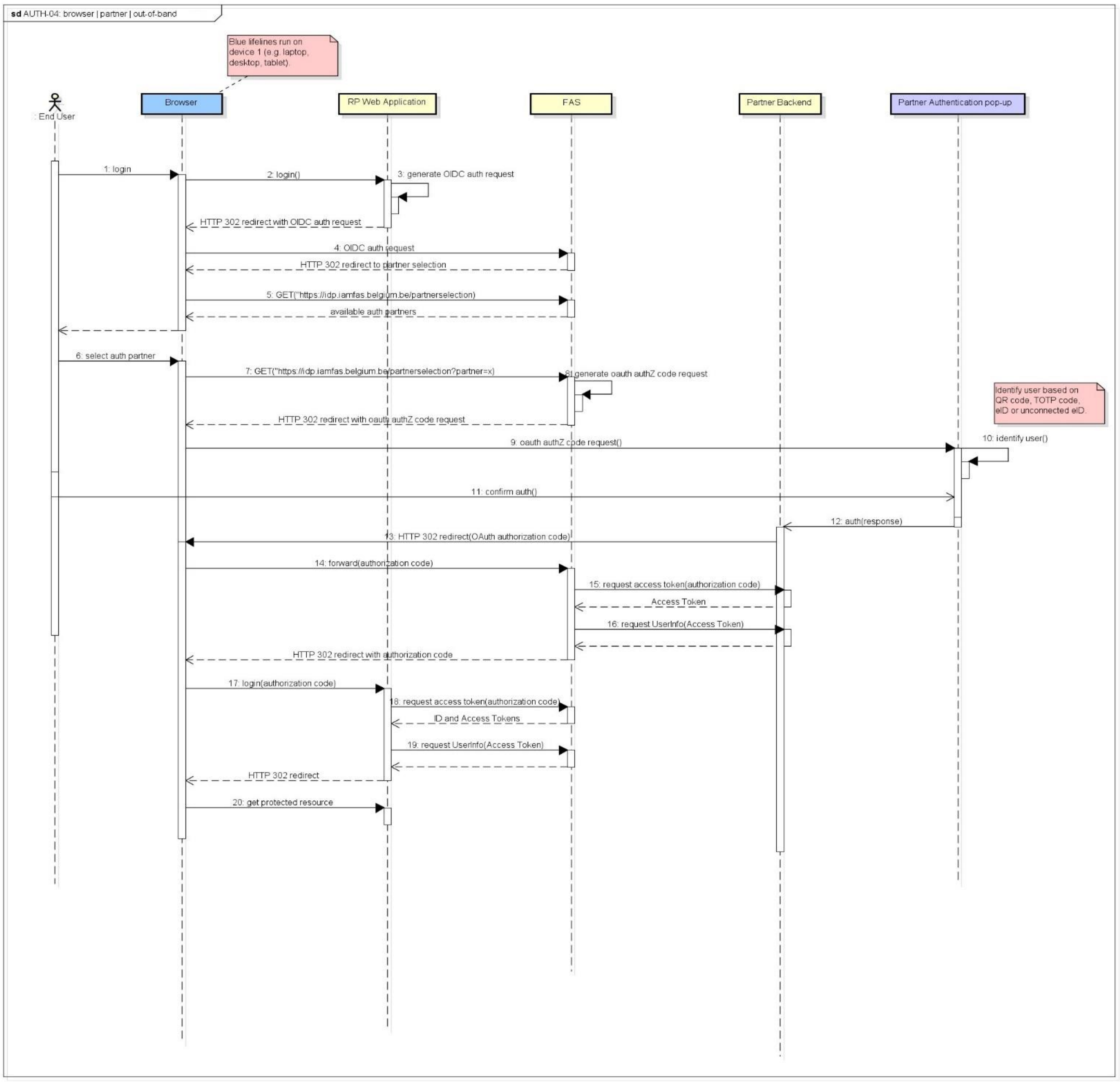
3.6.2 In-band: web application on mobile

This use case is quite similar to the previous use case. The only difference is that the process starts in a browser on the mobile device. The web intent, universal linking and the push notification mechanisms are still applicable.

3.6.3 Out-of-band: web eGov application on any device

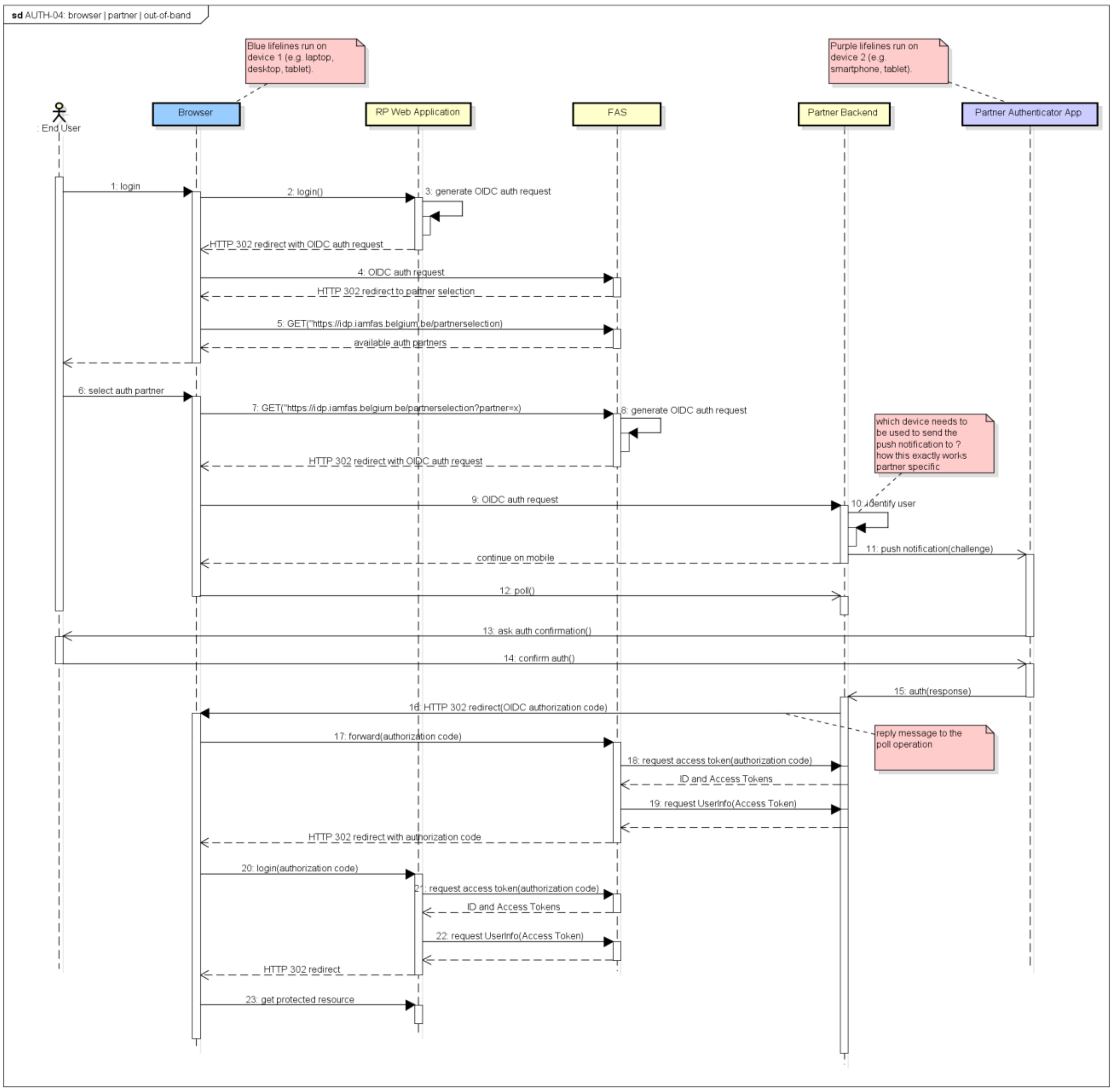
Example 1: mobile QR code scan

- The end user clicks onto the 'Log in with FAS' button in the web eGov application.
- The web eGov application redirects the end user to the FAS login page (Partner selection).
- The end user selects *Partner*.
- The FAS redirects the browser to the Partner login page.
- The Partner's system displays a QR code (or TOTP or eID).
- The end user scans the QR code with the Partner authenticator app (from his smartphone).
- The Partner system verifies the QR code after entering for example a PIN or a fingerprint and upon success considers the end user as successfully authenticated.
- The Partner system redirects the browser to the FAS.
- The FAS authenticates the end user and sends a browser redirect to the web eGov application.
- The FAS relays the received identity information to the web eGov application.
- The end user is now authenticated in the web eGov application.


sd AUTH-04: browser | partner | out-of-band
Blue lifelines run on device 1 (e.g. laptop, desktop, tablet).
: End User
Browser
RP Web Application
FAS
Partner Backend
Partner Authentication pop-up
1: login
2: login()
3: generate OIDC auth request
HTTP 302 redirect with OIDC auth request
4: OIDC auth request
HTTP 302 redirect to partner selection
5: GET("https://idp.iamfas.belgium.be/partnerselection)
available auth partners
6: select auth partner
7: GET("https://idp.iamfas.belgium.be/partnerselection?partner=x)
8: generate oauth authZ code request
HTTP 302 redirect with oauth authZ code request
Identify user based on QR code, TOTP code, eID or unconnected eID.
9: oauth authZ code request()
10: identify user()
11: confirm auth()
12: auth(response)
13: HTTP 302 redirect(OAuth authorization code)
14: forward(authorization code)
15: request access token(authorization code)
Access Token
16: request UserInfo(Access Token)
HTTP 302 redirect with authorization code
17: login(authorization code)
18: request access token(authorization code)
ID and Access Tokens
19: request UserInfo(Access Token)
HTTP 302 redirect
20: get protected resource

Example 2: push notification

- The end user clicks on the 'Log in with FAS' button in the web eGov application.
- The web eGov application redirects the browser to the FAS authentication Partner selection page.
- The end user selects *Partner*.
- The FAS redirects the end user to a Partner web page.
- The Partner's site asks the end user to fill in his/her identification/telephone number to receive a push notification.
- The end user opens the push notification on his/her smartphone.
- The end user authenticates, e.g. by entering PIN or using fingerprint on the mobile authenticator app.
- The Partner system verifies the provided credentials and upon success considers the end user as successfully authenticated.
- Partner sends the end user back to the FAS.
- The FAS relays the received identity information to the web eGov application.
- The end user is now authenticated in the web eGov application.


sd AUTH-04: browser | partner | out-of-band
Blue lifelines run on device 1 (e.g. laptop, desktop, tablet).
Purple lifelines run on device 2 (e.g. smartphone, tablet).
: End User
Browser
RP Web Application
FAS
Partner Backend
Partner Authenticator App
1: login
2: login()
3: generate OIDC auth request
HTTP 302 redirect with OIDC auth request
4: OIDC auth request
HTTP 302 redirect to partner selection
5: GET("https://idp.iamfas.belgium.be/partnerselection)
available auth partners
6: select auth partner
7: GET("https://idp.iamfas.belgium.be/partnerselection?partner=x)
8: generate OIDC auth request
which device needs to be used to send the push notification to ? how this exactly works partner specific
HTTP 302 redirect with OIDC auth request
9: OIDC auth request
10: identify user
11: push notification(challenge)
continue on mobile
12: poll()
13: ask auth confirmation()
14: confirm auth()
15: auth(response)
16: HTTP 302 redirect(OIDC authorization code)
reply message to the poll operation
17: forward(authorization code)
18: request access token(authorization code)
ID and Access Tokens
19: request UserInfo(Access Token)
HTTP 302 redirect with authorization code
20: login(authorization code)
21: request access token(authorization code)
ID and Access Tokens
22: request UserInfo(Access Token)
HTTP 302 redirect
23: get protected resource

4 REFERENCES

4.1 NORMATIVE REFERENCES

- [RFC 2119] Key words for use in RFCs to Indicate Requirement Levels: https://tools.ietf.org/html/rfc2119, Mars 1997

OAuth 2.0 Core

- [RFC 6749] OAuth 2.0 Framework : https://tools.ietf.org/html/rfc6749, October 2012
- [RFC 6750] Bearer Token Usage : https://tools.ietf.org/html/rfc6750, October 2012
- [RFC 6819] Threat Model and Security Considerations : https://tools.ietf.org/html/rfc6819, January 2013

OAuth 2.0 Extensions

- OAuth 2.0 Device Flow for Browserless and Input Constrained Devices: https://tools.ietf.org/pdf/draft-ietf-oauth-device-flow-06.pdf, May 2017, expires December 2017
- [RFC 7662] OAuth 2.0 Token Introspection : https://tools.ietf.org/html/rfc7662, October 2015. To determine the active state and meta-information of a token
- [RFC 7636] Proof Key Proof Key for Code Exchange by Oauth Public Clients : https://tools.ietf.org/html/rfc7636, September 2015 (Proposed Standard)
- [RFC 8252] OAuth 2.0 for native apps: https://tools.ietf.org/html/rfc8252, October 2017
- [RFC 7519] JSON Web Token : http://tools.ietf.org/html/rfc7519, May 2015
- [RFC 7521] OAuth Assertions Framework : https://tools.ietf.org/html/rfc7521, May 2015
- [RFC 7522] SAML2 Bearer Assertion : https://tools.ietf.org/html/rfc7522, May 2015. For integrating with existing identity systems
- [RFC 7523] JWT Bearer Assertion : https://tools.ietf.org/html/rfc7523, May 2015. For integrating with existing identity systems

OpenID Connect 1.0 specification

The OpenID Connect 1.0 specification consists of these documents:

- [Core] http://openid.net/specs/openid-connect-core-1_0.html , November 2014. Defines the core OpenID Connect functionality: authentication built on top of OAuth 2.0 and the use of Claims to communicate information about the end user
- [Discovery] http://openid.net/specs/openid-connect-discovery-1_0.html, November 2014.- Optional, defines how Clients dynamically discover information about OpenID Providers
- [Dynamic Registration] http://openid.net/specs/openid-connect-registration-1_0.html, November 2014- Optional, defines how clients dynamically register with OpenID Providers
- [OAuth 2.0 Multiple Response Types] http://openid.net/specs/oauth-v2-multiple-response-types-1_0.html, February 2014. Defines several specific new OAuth 2.0 response types
- [OAuth 2.0 Form Post Response Mode]- http://openid.net/specs/oauth-v2-form-post-response-mode-1_0.html, April 2015. Optional, defines how to return OAuth 2.0 Authorization Response parameters (including OpenID Connect Authentication Response parameters) using HTML form values that are auto-submitted by the User Agent using HTTP POST

- [Session Management] http://openid.net/specs/openid-connect-session-1_0.html, January 2017. Optional, defines how to manage OpenID Connect sessions, including postMessage-based logout functionality
- [Front-Channel Logout] – http://openid.net/specs/openid-connect-frontchannel-1_0.html, January 2017. Optional, defines a front-channel logout mechanism that does not use an OP iframe on RP pages
- [Back-Channel Logout] – http://openid.net/specs/openid-connect-backchannel-1_0.html, January 2017. Optional, defines a logout mechanism that uses back-channel communication between the OP and RPs being logged out

eIDAS documents

- [eIDAS] EU Regulation № 910/2014 on electronic identification and trust services for electronic transactions in the European internal market : https://ec.europa.eu/digital-single-market/en/policies/trust-services-and-eidentification
- Commission Implementing Regulation (EU) 2015/1502 of 8 September 2015 on setting out minimum technical specifications and procedures for assurance levels for electronic identification means : http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:JOL_2015_235_R_0002
- Set of eIDAS-compliant technical specifications : https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eIDAS+eID+Profile

4.2 NON-NORMATIVE REFERENCES

- **OpenID Connect 1.0 -** Basic Client Implementer's Guide : http://openid.net/specs/openid-connect-basic-1_0.html, August 2015. Simple subset of the Core functionality for a web-based Relying Party using the OAuth code flow

- **ISACA Glossary:** http://www.isaca.org/About-ISACA/History/Documents/ISACA-Glossary-English-French_mis_Fra_0615.pdf, 2015.

- **Control framework for assessing assurance level of the electronic identification service:** Self-evaluation tool developed by the FPS BOSA - DG Digital Transformation/Domain Identification, Authentication and Authorisation, October 2017. Tool for evaluating the conformity of the electronic identification service to the required assurance levels as defined in the Commission Implementing Regulation (EU) 2015/1502 of 8 September 2015 on setting out minimum technical specifications and procedures for assurance levels for electronic identification means. Available on demand via servicedesk.dto@bosa.fgov.be.