Federale Overheidsdienst Binnenlandse Zaken
Algemene Directie Instellingen en Bevolking
Rijksregister
Dienst Klantenbeheer

Aan het College van burgemeester en Schepenen
Ter attentie van de dienst Bevolking

Aan de gebruikers van het Rijksregister

Aan de verstrekkers van informaticadiensten

Uw contactpersoon	**T**	**Uw referentie**	**Bijlagen**
Jean Cuvelier	02 518 22 90		3
E-mail	**F**	**Onze referentie**	**Brussel**
Jean.cuvelier@rrn.fgov.be	02 518 27 90	III	4 november 2020

Rijksregister van de natuurlijke personen. - Overstap naar de 3de generatie webservices.

Mevrouwen,
Mijne heren,

Met deze nota willen wij u op de hoogte brengen van de ontwikkelingen en evolutie op het vlak van de webservices van het Rijksregister.

1. Context.

De diensten van het Rijksregister worden voortdurend gemoderniseerd om een optimaal kwalitatieve en veilige communicatie met onze webgebruikers te garanderen.

De tweede generatie webservices verscheen in 2012 en dat betekende het definitieve einde van het X25-protocol dat gedurende tientallen jaren werd gebruikt. Die tweede generatie heeft het toen mogelijk gemaakt om nieuwe manieren van toegangsbeheer (NN, CN, EN) te gebruiken, wat authentificatie met behulp van het ondernemingsnummer toeliet (in het kader van de vertrouwenscirkels), naast authentificatie met het Rijksregisternummer.

Door de evolutie op het gebied van technologie en veiligheid kan de 2de generatie nu plaatsmaken voor de 3de generatie, die vanaf nu beschikbaar en operationeel is om te gebruiken bij alle diensten van het Rijksregister. De twee generaties worden op dit moment al samen gebruikt.

In een verdere toekomst en met het oog op het voorbereiden van de evolutie naar de vierde generatie webservices van het Rijksregister, zal de 2de generatie op 1 juli 2023 verdwijnen en zal op dat moment enkel de derde generatie worden gebruikt.

Daarna zal de 3de generatie vervangen worden, wat voorzien is tegen 2028.

2. Eigenschappen van de 3de generatie webservices.

- Alle functionaliteiten van de 2de generatie blijven behouden.

- Verbeteringen in vergelijking met de vorige generatie:

  - Betreffende de normen en de interoperabiliteit: het SOAP-protocol voor webservices wordt vervangen door het REST-protocol voor webservices (of RESTful). Dat laatste protocol is recenter en flexibeler.

  - Wat de veiligheid betreft, aanvaarden de webservices nu EC-certificaten (elliptic curve), naast de RSA-certificaten. Daardoor verbetert de veiligheid.

  - Wat de opvolging (tracing) van het antwoord op een verzoek betreft, kan er nu een specifiek identificatiekenmerk worden opgenomen in het verzoek. Deze laatste zal ook worden meegedeeld bij het antwoord op het verzoek en maakt het mogelijk een link te leggen tussen de vraag en het antwoord.

  - Wat het verzoek betreft, kan de boodschap die naar het Rijksregister wordt gestuurd, ondertekend worden met een gebruikerscertificaat van een natuurlijk of een rechtspersoon. Dat is vooral verplicht wanneer een bijwerkingstransactie via een dienstenintegrator gaat. Voor de wijzigingen van de toegangsrechten via webservices moet de transactie ook worden ondertekend.

3. Praktisch: hoe veranderen naar de nieuwe generatie webservices?

U kunt de verandering van het SOAP-protocol naar het REST-protocol en de nieuwe functionaliteiten laten uitvoeren door uw informaticadienstenverstrekker of uw eigen informaticadienst.

Wat het Rijksregister betreft, is er niet systematisch een interventie nodig, behalve in geval van verandering van certificaat of van de koppeling ervan.

In de bijlagen vindt u de volgende documentatie:

- het document RESTful API "in the Swagger Specification" van de REST services van het RR;
- de documentatie van de veranderingen die zijn aangebracht aan de bestaande services (waaronder de nieuw te gebruiken URL's);
- voorbeelden van tests van een API REST met behulp van Curl.

Er is een testomgeving (8020) opgezet. Die kan worden gebruikt voor de voorbereiding van uw overstap naar de 3de generatie.

Park Atrium
Koloniënstraat 11 T 02 518 21 16 helpdesk.belpic@rrn.fgov.be
1000 Brussel F 02 518 26 16 www.ibz.rrn.fgov.be

4. Met wie kunt u contact opnemen voor bijkomende inlichtingen of hulp? :

Contactpersonen bij technische vragen :

- El Habib El Meskioui, 02/518.21.33 - ElHAbib.ElMeskioui@rrn.fgov.be
- Paquito Bartiaux, 02/518.23.27 - Paquito.Bartiaux@rrn.fgov.be
- Ives Gobau, 02/518.21.05 - Ives.Gobau@rrn.fgov.be
- Nicolas Hermel, 02/518.22.39 - Nicolas.Hermel@rrn.fgov.be
- Helpdesk : 02/518.21.16 - Helpdesk.Belpic@rrn.fgov.be

Met hoogachting,

Jacques Wirtz (Signature)
Signature numérique de Jacques Wirtz (Signature)
Date : 2020.11.05 12:32:36 +01'00'

Jacques Wirtz,
Directeur-generaal

Park Atrium
Koloniënstraat 11 T 02 518 21 16 helpdesk.belpic@rrn.fgov.be
1000 Brussel F 02 518 26 16 www.ibz.rrn.fgov.be

RRN REST WS 3de generatie

1. Doel van dit document

Dit document beschrijft de migratie van de RRN Web services in SOAP 2e generatie, naar de RRN Web services REST 3e generatie.

2. Toegang tot web services

De beveiliging van de web services blijft dezelfde zoals in de SOAP versie van 2e generatie, toegepast op het niveau van het transport, en daarna op het niveau van de boodschap. Bij het raadplegen van het RRN, zijn de transacties tussen de toepassing van de klant en de web service beveiligd met het TLS protocol, de toepassing van de klant wordt geauthenticeerd met een X509 certificaat.

3. Overschakelen van het SOAP-model naar het REST-model

De aanpak is om een reeks bedrijfsactiviteiten te projecteren op een resource-georiënteerde architectuur. De huidige web services zijn samengesteld uit 2 SOAP webservices: de “SAML-generator” en de “Transactie uitvoerder.” Deze twee services bevatten elk één methode. Elk van beide bevat een handtekening, en, indien nodig, uitzonderingen in het geval van een fout. Deze methodes worden vervangen door interactieve modellen die toelaten om functionaliteit toe te voegen aan de bedrijfsapplicatie. De beschrijving van de interacties is gedefinieerd in de OpenAPI documenten die samen met dit document geleverd worden. Deze OpenAPI documenten kunnen vervolgens gebruikt worden met applicaties die automatisch documentatie aanmaken voor het visualiseren van de API, en met ontwikkelsoftware die automatisch code genereren voor het maken van applicaties in diverse programmeertalen.

4. Handtekening voor transacties die gegevens aanpassen

Bovenop de beschikbare aanvullende parameters (zie openAPI documentatie) in de REST versie van de RRN webservices, laat deze versie ook toe dat transacties die aanpassingen doen voorzien zijn van een handtekening. De handtekening op de aanpassende transacties is verplicht voor alle gebruikers die indirect toegang hebben tot het RRN via een leverancier van RRN diensten.

4.1. Transactie zonder handtekening

- De parameter « txMessage » blijft ongewijzigd zoals in de SOAP versie,
- De parameter « signed » bevat de waarde « 00 »
-

4.2. Transactie met handtekening

- De parameter « txMessage » wordt vervangen door « signedMessage »,
- De parameter « signed » bevat de waarde « 01 »,
- De parameter « signedMessage » bevat de samenvoeging van diverse delen van de JWS, zijnde een hoofding (header,) de inhoud (payload) en de handtekening (signature,) gecodeerd in het formaat BASE64 en onderscheiden van elkaar met een punt (zoals beschreven in RFC7515 : *the JWS Compact Serialization*[1]).
- Element <header> van signedMessage
    - Parameter die het gebruikte handtekening algoritme aangeeft
        - alg : één van de volgende waarden (RS256, RS384, RS512)
    - Parameter noodzakelijk voor de verificatie via RSA algoritme
        - x5c : Certificaat of keten van certificaten die overeenkomen met de sleutel die gebruikt werd bij het maken van de handtekening op één regel.
- Element <payload> van signedMessage
    - Deze parameter hergebruikt het element « txMessage » van de originele opvraging
    - Ze bevat een variabel element dat dat uniek is voor elke transactie (cnonce : client number used once) te bepalen door de klant in samenwerking met de leverancier.
      Voorbeeld : {"txMessage":"%XM25 RRN000---------00000C0----------
      00500011111000000----------NH",
      "cnonce":"6d8a3402-34f3-11e9-b210-d663bd873d93"}
    - Dit alles wordt in zijn geheel geëncodeerd in Base 64.
- Element <signature> van signedMessage
    - Bevat het resultaat van de handtekening van de samenvoeging van « Header » en « Payload » en van elkaar gescheiden door een punt « . » gebruik makende van het gekozen algoritme (alg) en de geheime sleutel van het certificaat (x5c). Het geheel is dan geëncodeerd in Base 64.

[1] https://www.rfc-editor.org/rfc/pdfrfc/rfc7515.txt.pdf (chapitre 7.1)

5. URL voor RRN REST web services

5.1. Test omgeving

SAML aanmaken :
https://www.webtcp20.rrn.fgov.be:5443/rrnws/rest/v3.0/assertions

Transactie uitvoeren :
https://www.webtcp20.rrn.fgov.be:5443/rrnws/rest/v3.0/transactions

5.2. Productie omgeving

SAML aanmaken :
https://www.rrnass.rrn.fgov.be:9920/rrnws/rest/v3.0/assertions

Transactie uitvoeren :
https://www.rrnass.rrn.fgov.be:9920/rrnws/rest/v3.0/transactions

6. Voorbeeld voor het gebruiken van Webservices 3 met Curl

6.1. Parameters voor het commando curl

- **CACERTFILE**
  → referentie naar het bestand dat de volledige keten van CA certificaten van de server bevat.

- **saml-parameters.json**
  → referentie naar het bestand dat de verschillende parameters bevat voor het verkrijgen van de SAML Assertion. (policy, certificat, requestId):
  Voorbeeld :

```
{
    "policy":"CN_POLICY",
    "certificate":"BASE64_CERTIFICATE",
    "requestId":"12345678-1234-1234-1234-123456789012"
}
```

  BASE64_CERTIFICATE moet hier vervangen worden door de BASE64 geencodeerde PEM van het certificaat. Ter verduidelijk, het certificaat in PEM versie, inclusief BEGIN en END headers moet geencodeerd worden in BASE64.

- **transaction-parameters.json**
  → referentie naar het bestand dat de verschillende parameters bevat voor het uitvoeren van een transactie. (txMessage, signed, signedMessage, rrnDestination, requestId) :
  Voorbeeld :
  Niet gehandtekend

```
{
    "txMessage":"%WS25 RRN0007C906206300000C07906011259105000100000021867906011259 1 F",
    “signed":"00",
    "rrnDestination":"RRN_WS",
    "requestId":"12345678-1234-1234-1234-123456789012"
```

Gehandtekend :

```
{
        "signedMessage":"eyJhbGciOiJSUzI1NiIsIng1YyImcwSlcw……..",
        "signed":"01",
        "rrnDestination":"RRN_WS",
        "requestId":"12345678-1234-1234-1234-123456789012"
}
```

Indien de parameter "signed" ontbreekt, dan wordt er gewerkt zonder handtekening.

- **CLIENT_CERTIFICATE**
  → Referentie naar het bestand dat de geheime sleutel en bijhorend certificaat bevat in PEM formaat.

6.2. Aanvraag van SAML Assertion

curl --location --connect-timeout 15 --max-time 15 --cacert $CACERTFILE --data-binary @saml-parameters.json --header "Expect:" --header "Content-type: application/json" --cert $CLIENT_CERTIFICATE --url https://www.webtcp20.rrn.fgov.be:5443/rrnwa/rest/v3.0/assertions > samlAssertion8020

6.2.1. Aanpassen van gegenereerd bestand (samlAssertion8020)

Het antwoord is weggeschrven in het bestand samlAsserion8020 in het JSON formaat en bevat 2 parameters, “assertion” en “requestid”.
Enkel de SAML assertion moet bewaard blijven, zonder de naam van de parameter, en voorafgegaan door de prefix “Bearer “ (met spatie tussen Bearer en de Saml assertion.
Deze kan voortaan gebruikt worden voor het uitvoeren van transacties.

6.3. Uitvoering van de transactie

curl --location --connect-timeout 15 --max-time 15 --cacert $CACERTFILE --header "Content-type: application/json" --header "Authorization: $(cat samlAssertion8020)" --data-binary @transaction-parameters.json --cert $CLIENT_CERTIFICATE --url https://www.webtcp20.rrn.fgov.be:5443/rrnws/rest/v3.0/transactions

```
openapi: 3.0.0
info:
    description: ""
    version: "3.0.0"
    title: "Swagger RRN REST SAML generator."
servers:
  - url: https://www.webtcp20.rrn.fgov.be:5443/rrnws/rest/v3.0
    description:  Main (Test Env 8020) server
paths:
  /transactions:
    post:
      summary: "Execute a given transaction."
      description: ""
      security:
        - bearerAuth: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                requestId:
                  type: string
                txMessage:
                  type: string
                signedMessage:
                  type: string
                signed:
                  type: string
                rrnDestination:
                  type: string
                options:
                  type: string
                txOutputMessage:
                  type: string
                  description: for futur use.
      responses:
        '200':
          description: A Transaction response
          content:
            application/json:
              schema:
                type: object
                properties:
                  requestId:
                    type: string
                    description: The request ID.
                  transactionResponse:
                    type: string
                    description: The transaction response.
                  signed:
                    type: string
                    description: The flag signed/unsigned request.
                  rrnDestination::
                    type: string
```

```
                    description: The RRN Destiniation given in the
request.
                  options:
                    type: string
                    description: Options as given in the request.
                  txOutputMessage:
                    type: string
                    description: for futur use.

# 1) Define the security scheme type (HTTP bearer)
components:
  securitySchemes:
    bearerAuth:            # arbitrary name for the security scheme
      type: http
      scheme: bearer
      bearerFormat: Base64 SAML token    # optional, arbitrary value
for documentation purposes
# 2) Apply the security globally to all operations
security:
  - bearerAuth: []         # use the same name as above
```

```
openapi: 3.0.0
info:
    description: ""
    version: "3.0.0"
    title: "Swagger RRN REST SAML generator."
servers:
  - url: https://www.webtcp20.rrn.fgov.be:5443/rrnws/rest/v3.0
    description:  Main (Test Env 8020) server

paths:
  /assertions:
    post:
      summary: "Creates a token SAML."
      description: ""
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                requestId:
                  type: string
                policy:
                  type: string
                certificate:
                  type: string
      responses:
        '200':
          description: A SAML token
          content:
            application/json:
              schema:
                type: object
                properties:
                  requestId:
                    type: string
                    description: The request ID.
                  assertion:
                    type: string
                    description: The SAML token.
```