Änderungen in gemSpec_Krypt

In A_24658 werden die Pfadangaben gegen operationid(s) der OpenApi des Authorization-Service ausgetauscht. Die Claims des JWT werden als Mindestanforderung formuliert, der Mediatype in "application/json" geändert.

alt:

A_24658 - VAU-Protokoll: PKI-basierte Client-Authentisierung
Eine VAU-Instanz MUSS über einen inneren HTTP-Request zwei Operation wie folgt anbieten.

(1) Über den Pfadnamen /FrischeParameter MUSS die VAU-Instanz Daten per HTTP-GET bereitstellen.
Diese Daten sind eine base64-kodierte Zeichenkette. Der Media-Type (Context-Type) MUSS 'text/plain;charset=UTF-8' sein.
Die Zeichenkette ist für einen Client opak. Diese Zeichenkette MUSS es einer VAU-Instanz ermöglichen, zu ermitteln, ob und wann ein VAU-HSM oder die Befugnisverifikations-VAU (BV-VAU [gemSpec_Aktensystem#3.4 Befugnisverifikations-Modul], das/die mit der VAU-Instanz verbunden ist, diese Zeichenkette erzeugt hat (bspw. Unix-Zeit + ECDSA-Signatur HSM). Es MUSS sichergestellt sein, dass nur das VAU-HSM oder die BV-VAU, das/die mit der VAU-Instanz verbunden ist, den Frischeparameter erzeugt haben kann.

(2) Über den Pfadnamen /PKI-Aut MUSS sie die Möglichkeit anbieten, per HTTP-POST ein Authentisierungstoken an die VAU-Instanz zu senden.

Die Authentisierungstoken sind JWT [RFC-7519], deren Body wie folgt aufgebaut ist:

```
{
    "type" : "ePA-Authentisierung über PKI",
    "iat" : ...zeit...,
    "challenge" : Frischeparameter,
    "sub": ...Telematik-ID...
}
```

Im Header des JWT MUSS innerhalb eines x5c-Array das "signierende" Zertifikat enthalten sein. Bei "sub" trägt der Client/Nutzer seine Telematik-ID ein.

Die VAU-Instanz MUSS das vom Client gesendete Authentisierungstoken prüfen:

1. Ist das im Header aufgeführte "signierende" Zertifikat gültig, inkl. OCSP-Prüfung (vgl. OCSP-Response-Caching nach [gemSpec_PKI#A_23225]).
2. Ist die Signatur valide.
3. Stimmt der "sub"-Wert im Body mit der Telematik-ID im "signierenden" Zertifikat überein.
4. Ist die "iat"-Zeit nicht älter als 10 Minuten.
5. Ist die "challenge" vom VAU-HSM oder einer BV-VAU erzeugt worden.
6. Ist die challenge nicht älter als 10 Minuten.
7. Ist der "type" gleich "ePA-Authentisierung über PKI"

Bei Prüfung mit positivem Prüfergebnis MUSS die VAU-Instanz den neuen Authentisierungsstatus (bspw. E-Rezept-FD wurde authentifiziert) in den Metadaten den Verbindungsschlüssel (K2_c2s_app_data, K2_s2c_app_data) vermerken.

Im inneren HTTP-Response-Header MUSS die VAU-Instanz das Nutzerpseudonym (vgl.

A_24770-*) übertragen.

Die VAU MUSS die Authentisierung beliebiger Nutzer mit gültigen AUT-Zertifikat und Telematik-ID darin erlauben (also nicht nur eingeschränkt auf den E-Rezept-FD).**[**<=, Aktensystem_ePA, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

neu:

A_24658-01 - VAU-Protokoll: PKI-basierte Client-Authentisierung
Eine VAU-Instanz MUSS über einen inneren HTTP-Request zwei Operation wie folgt anbieten.

(1) Über operationid = getFreshnessParameter MUSS die VAU-Instanz Daten per HTTP-GET bereitstellen.
Diese Daten sind eine base64-kodierte Zeichenkette. Der Media-Type (Context-Type) MUSS 'application/json'sein.
Die Zeichenkette ist für einen Client opak. Diese Zeichenkette MUSS es einer VAU-Instanz ermöglichen, zu ermitteln, ob und wann ein VAU-HSM oder die Befugnisverifikations-VAU (BV-VAU [gemSpec_Aktensystem_ePAfueralle#3.4 Befugnisverifikations-Modul], das/die mit der VAU-Instanz verbunden ist, diese Zeichenkette erzeugt hat (bspw. Unix-Zeit + ECDSA-Signatur HSM). Es MUSS sichergestellt sein, dass nur das VAU-HSM oder die BV-VAU, das/die mit der VAU-Instanz verbunden ist, den Frischeparameter erzeugt haben kann.

(2) Über operationid = sendAuthorizationRequestBearerToken MUSS sie die Möglichkeit anbieten, per HTTP-POST ein Authentisierungstoken an die VAU-Instanz zu senden.

Die Authentisierungstoken sind JWT [RFC-7519], deren Body mindestens folgende Elemente enthalten MUSS:

```
{
    "type" : "ePA-Authentisierung über PKI",
    "iat" : ...zeit...,
    "challenge" : Frischeparameter,
    "sub": ...Telematik-ID...
}
```

Im Header des JWT MUSS innerhalb eines x5c-Array das "signierende" Zertifikat enthalten sein. Bei "sub" trägt der Client/Nutzer seine Telematik-ID ein.

Die VAU-Instanz MUSS das vom Client gesendete Authentisierungstoken prüfen:

1. Ist das im Header aufgeführte "signierende" Zertifikat gültig, inkl. OCSP-Prüfung (vgl. OCSP-Response-Caching nach [gemSpec_PKI#A_23225]).
2. Ist die Signatur valide.
3. Stimmt der "sub"-Wert im Body mit der Telematik-ID im "signierenden" Zertifikat überein.
4. Ist die "iat"-Zeit nicht älter als 10 Minuten.
5. Ist die "challenge" vom VAU-HSM oder einer BV-VAU erzeugt worden.
6. Ist die challenge nicht älter als 10 Minuten.
7. Ist der "type" gleich "ePA-Authentisierung über PKI"

Bei Prüfung mit positivem Prüfergebnis MUSS die VAU-Instanz den neuen Authentisierungsstatus (bspw. E-Rezept-FD wurde authentifiziert) in den Metadaten den

Verbindungsschlüssel (K2_c2s_app_data, K2_s2c_app_data) vermerken.

Im inneren HTTP-Response-Header MUSS die VAU-Instanz das Nutzerpseudonym (vgl. A_24770-*) übertragen.

Die VAU MUSS die Authentisierung beliebiger Nutzer mit gültigen AUT-Zertifikat und Telematik-ID darin erlauben (also nicht nur eingeschränkt auf den E-Rezept-FD).**[**<=, Aktensystem_ePA, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

In A_25055 wird die Vorgabe der Rückgabe des VAU-NP im Header der Response entfernt, da dieses Element konkret im Responsebody an den Client übertragen wird.

alt:

A_25055 - VAU-Protokoll: OIDC Authentisierung oberhalb der VAU-Protokoll-Schicht
Eine VAU-Instanz MUSS für eine OIDC/OAuth2/PCKE notwendige HTTP Endpunkte per inneren HTTP-Request/Responses einem Client verfügbar machen (Codechallenge per GET zur Verfügung stellen, Authentication Code per POST vom Client empfangen) machen.

Nach erfolgreichen Authentisierung über OIDC MUSS die VAU-Instanz in den Metadaten der Verbindungsschlüssel (KeyID, K2_*_app_data) den neuen Authentisierungszustand vermerken (Rückwirkung auf A_24634-*).

Nach erfolgreicher Authentisierung über OIDC, genauer innerhalb des Response-Headers (innerer Request) nach dem POST-Request für den Authentication-Codes durch den Client, MUSS sie ein Nutzerpseudonym dem Client in der HTTP-Variable "VAU-NP" verfügbar machen, vgl. A_24770-*.**[**<=, Aktensystem_ePA, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

neu:

A_25055-01 - VAU-Protokoll: OIDC Authentisierung oberhalb der VAU-Protokoll-Schicht
Eine VAU-Instanz MUSS für eine OIDC/OAuth2/PCKE notwendige HTTP Endpunkte per inneren HTTP-Request/Responses einem Client verfügbar machen (Codechallenge per GET zur Verfügung stellen, Authentication Code per POST vom Client empfangen) machen.

Nach erfolgreichen Authentisierung über OIDC MUSS die VAU-Instanz in den Metadaten der Verbindungsschlüssel (KeyID, K2_*_app_data) den neuen Authentisierungszustand vermerken (Rückwirkung auf A_24634-*).

Nach erfolgreicher Authentisierung über OIDC, genauer innerhalb der Response (innerer Request) nach dem POST-Request für den Authentication-Codes durch den Client, MUSS sie ein Nutzerpseudonym dem Client in der HTTP-Variable "VAU-NP" verfügbar machen, vgl. A_24770-*.
[<=, Aktensystem_ePA, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

Änderungen in gemSpec_Aktensystem_ePAfueralle

In A_25165-02 werden die nicht benötigten Claims "oid" und "exp" entfernt.

alt:

A_25165-02 - Authorization Service: JWT Bearer-Token des E-Rezept-Fachdienstes
Das Authorization Service MUSS sicherstellen, dass die Authentifizierung des E-Rezept-Fachdienstes über die Schnittstelle `I_Authorization_Service` durch Verwendung eines gültig signierten JWT Bearer Token mit den dargestellten Mindest-Inhalten und Prüfung durch Regel 'rr0' des Befugnisverifikations-Moduls erfolgt. Die Claims in 'Payload' MÜSSEN dazu die Vorgaben aus [gemSpec_Krypt], A_24658* befolgen.

Part	Claim Name	Claim	Anmerkung
Protected Header			
	"typ"	"JWT"	
	"alg"	"ES256"	
	"x5c"	Signaturzertifikat C.FD.AUT	
Payload			
	"type"	"ePA-Authentisierung über PKI"	fester Wert
	"iat"	Zeitstempel Ausgabezeitpunkt	Beispiel: "1705674544"
	"exp"	Verfalldatum, = "iat" + 20 min	Beispiel: "1705675744"
	"challenge"	Frischeparameter	siehe [gemSpec_Krypt]
	"sub"	Telematik-ID des E-Rezept-Fachdienstes	
	"oid"	oid_erp-vau (1.2.276.0.76.4.25)	fester Wert gemäß [gemSpec_OID]

[<=, Aktensystem_ePA, eRp_FD, funkt. Eignung: Test Produkt/FA**]**

neu:

A_25165-03 - Authorization Service: JWT Bearer-Token des E-Rezept-Fachdienstes
Das Authorization Service MUSS sicherstellen, dass die Authentifizierung des E-Rezept-Fachdienstes über die Schnittstelle `I_Authorization_Service` durch Verwendung eines gültig signierten JWT Bearer Token mit den dargestellten Mindest-Inhalten und Prüfung durch Regel 'rr0' des Befugnisverifikations-Moduls erfolgt. Die Claims in 'Payload' MÜSSEN dazu die Vorgaben aus [gemSpec_Krypt], A_24658* befolgen.

Part	Claim Name	Claim	Anmerkung
Protected Header			
	"typ"	"JWT"	
	"alg"	"ES256"	
	"x5c"	Signaturzertifikat C.FD.AUT	
Payload			
	"type"	"ePA-Authentisierung über PKI"	fester Wert
	"iat"	Zeitstempel Ausgabezeitpunkt	Beispiel: "1705674544"
	"challenge"	Frischeparameter (freshness parameter)	siehe [gemSpec_Krypt]
	"sub"	Telematik-ID des E-Rezept-Fachdienstes	

[<=, Aktensystem_ePA, eRp_FD, funkt. Eignung: Test Produkt/FA**]**

Änderung in Kapitel 3.19.1 Übersicht der Schnittstellen des Aktensystems

Tabelle 37: Übersicht der Schnittstellen des Aktensystems

Hinzufügen der getFreshnessParameter Operation für I_Authorization_Service.yaml

Schnittstellen des Aktensystems (Nutzung nur bei etabliertem VAU-Kanal)	
.....	
I_Authorization_Service	
Schnittstelle der Autorisierung für den Login	
`getFHIRVZDtoken`	Diese Operation bezieht das search-access-token für den Zugriff auf den FHIR VZD vom Aktensystem
`getNonce`	Diese Operation bezieht einen eindeutigen einmalig generierten Zufallswert für die Client Attestierung
`sendAuthorizationRequestSC`	Diese Operation initiiert die Authentifizierung eines Leistungserbringers
`sendAuthorizationRequestFdV`	Diese Operation initiiert die Authentifizierung eines ePA-FdV
`getFreshnessParameter`	Diese Operation erzeugt einen Frischeparameter für die Authentisierung mittels Bearer Token
`sendAuthorizationRequestBearerToken`	Diese Operation initiiert die Authentifizierung über Bearer Token
`sendAuthCodeSC`	Diese Operationen übergibt den Authorization Code für den Bezug des ID-Token
`sendAuthCodeFdV`	Diese Operationen übergibt den Authorization Code für den Bezug des ID-Token
`logoutFdV`	Diese Operation beendet aktiv die Sitzung
....	

Änderungen in I_Authorization_Service.yaml

alt: (getChallenge)

```
  /epa/authz/v1/getChallenge:
    parameters:
```

```
    - $ref: '#/components/parameters/useragent'
  get:
    tags:
      - Authorization BearerToken
    operationId: getChallengeForBearerToken
    summary: (getChallengeForBearerToken) Get freshness parameter for a bearer
token
    externalDocs:
      description: 'eRP backend authentication: gemSpec_Krypt, chapter "7.4
Authentisierung des E-Rezept-FD als ePA-Client"'
      url: https://gemspec.gematik.de/docs/gemSpec/
    description: |-
    Get a new challenge (freshness parameter) for a new bearer token for the
authorization by bearer token.
     **Client**</br>
     The ePrescription backend shall use the challenge for a signed JWT (bearer token)
     according to requirement gemSpec_Aktensystem_ePAfuerAlle, A_25165*.
     **Provider**</br>
     The returned challenge shall follow the requirements in gemSpec_Krypt, A_24658*
and be
     verifiable by HSM rule 'rr0'.

        ........

            properties:
              challenge:
                description: A base64 encoded challenge for a bearer token (health record
system specific content)
                type: string
                format: byte
                example: YSBiYXNlNjQgZW5jb2RlZCBjaGFsbGV.....
```

neu (getFreshnessParameter):

```
/epa/authz/v1/freshness:
  parameters:
    - $ref: '#/components/parameters/useragent'
  get:
    tags:
      - Authorization BearerToken
    operationId: getFreshnessParameter
    summary: (getFreshnessParameter) Get freshness parameter for a bearer token
    externalDocs:
      description: 'eRP backend authentication: gemSpec_Krypt, chapter "7.4
Authentisierung des E-Rezept-FD als ePA-Client"'
      url: https://gemspec.gematik.de/docs/gemSpec/
    description: |-
    Get a new freshness parameter for a new bearer token for the authorization by
bearer token.
     **Client**</br>
     The ePrescription backend shall use thefreshness parameter for a signed JWT
(bearer token)
     according to requirement gemSpec_Aktensystem_ePAfuerAlle, A_25165*.
```

```
      **Provider**</br>
      The returned freshness parameter shall follow the requirements in gemSpec_Krypt,
A_24658* and be
      verifiable by HSM rule 'rr0'.

        ........

            properties:
              freshness:
                description: A base64 encoded freshness parameter for a bearer token
(health record system specific content)
                type: string
                format: byte
                example: YSBiYXNlNjQgZW5jb2RlZCBjaGFsbGV.....
```

alt (sendAuthorizationRequestBearerToken):

```
  /epa/authz/v1/send_authorization_request_bearertoken:
    parameters:
      - $ref: '#/components/parameters/useragent'
    post:
      tags:
        - Authorization BearerToken
      operationId: sendAuthorizationRequestBearerToken
      summary: (sendAuthorizationRequestBearerToken) Client authorization based on
JWT authorization grant.
      externalDocs:
        description: 'BearerToken: gemSpec_Aktensystem_ePAfuerAlle, chapter "3.15.3
Anforderungen an den Authorization Service für die Authentisierung des E-Rezept-
Fachdienstes"'
        url: https://gemspec.gematik.de/docs/gemSpec/
      description: |-
        Authorization of the ePrescription backend (E-Rezept-Fachdienst).
        **Client**</br>
        The ePrescription backend shall send a signed JWT (bearerToken) according
        to requirement gemSpec_Aktensystem_ePAfuerAlle, A_25165*.
        The token shall contain a fresh challenge (see: _getChallengeForBearerToken_).
        **Provider**</br>
        The received token shall be validated with HSM rule 'rr0'. The resulting HSM-ID-
Token shall be added to the user session.
        The received token shall have a claim "sub", this claim shall state the telematik-id
of the ePrescription backend.
        The received token shall have a claim 'exp' indicating a point in time in the future of
'current time'.
```

.

neu (sendAuthorizationRequestBearerToken):

```
  /epa/authz/v1/send_authorization_request_bearertoken:
    parameters:
      - $ref: '#/components/parameters/useragent'
    post:
      tags:
        - Authorization BearerToken
```

```
    operationId: sendAuthorizationRequestBearerToken
    summary: (sendAuthorizationRequestBearerToken) Client authorization based on
JWT authorization grant.
    externalDocs:
      description: 'BearerToken: gemSpec_Aktensystem_ePAfuerAlle, chapter "3.15.3
Anforderungen an den Authorization Service für die Authentisierung des E-Rezept-
Fachdienstes"'
      url: https://gemspec.gematik.de/docs/gemSpec/
    description: |-
      Authorization of the ePrescription backend (E-Rezept-Fachdienst).
      **Client**</br>
      The ePrescription backend shall send a signed JWT (bearerToken) according
      to requirement gemSpec_Aktensystem_ePAfuerAlle, A_25165*.
      The token shall contain a freshness parameter (see: _getFreshnessParameter_).
      **Provider**</br>
      The received token shall be validated with HSM rule 'rr0'. The resulting HSM-ID-
Token shall be added to the user session.
      The received token shall have a claim "sub", this claim shall state the telematik-id
of the ePrescription backend.
      <line removed>
```

Änderungen in schemas:

```
  BearerTokenType:
    description: |
      "A JSON Web Token (JWT) with following format build according to RFC-
7515:</br>
      base64url (protected_header) + '.' + base64url (payload) + '.' + base64url
(signature)"</br>
      Content for ePrescription backend bearerToken</br>
      - protected_header contains:
        - "typ": "JWT"
        - "alg": "ES256"
        - "x5c": signature certificate c.fd.aut
      - payload claims:
        - "type": "ePA-Authentisierung über PKI" (fixed value)
        - "iat": issued at timestamp
        - <line removed>
        - "challenge": freshness parameter (base64 encoded)
        - "sub": Telematik-ID ePrescription backend
        - <line removed>
      - signature: contains token signature
```