FDZ

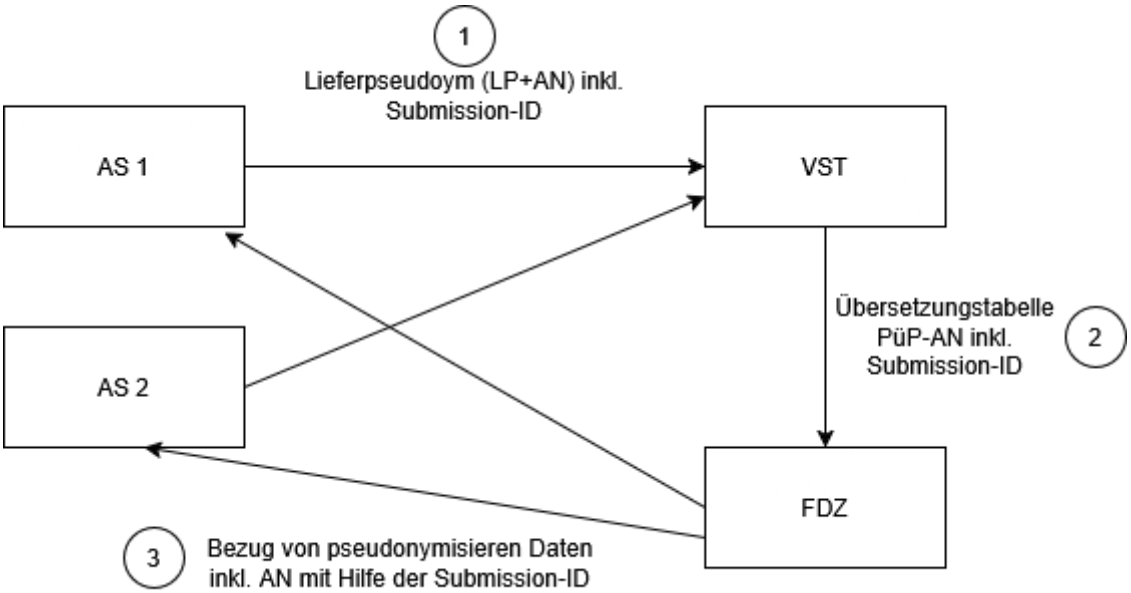
FDZ

Einleitung

Das FDZ tritt in zwei Rollen auf -- einmal als Server und einmal als ePA-Client.

Wichtigste Anwendungsfälle sind:

- Annahme von Übersetzungstabelle PüP-AN (Submission mit Submission-ID) von der VST (FDZ ist Server)
- Abholen von pseudonymisierten Daten inkl. AN (Submission mit Submission-ID) vom ePA-Aktensystem
  (FDZ ist "normaler" ePA-Client)



Ein AS erzeugt zufällig eine Submission-ID und stellt eine Submission. Dabei werden vom AS Auftragsnummer (AN), Lieferpseudonyme (LP) und pseudoynmisierte Daten erzeugt.

Die Submission-ID (256-Bit-Wert) ist ein wichtiger Zuordnungsschlüsselwert (im Datenbanksinne / Zuordnungsschlüssel).

Das AS sendet eine Abbildungstabelle LP-AN inkl. Submission-ID an die VST. (Link)

Die VST sendet eine Abbildungstabelle PüP-AN inkl. Submission-ID an das FDZ. (Link)

Das FDZ bezieht als "normaler" ePA-Client vom AS die pseudonymisierten Daten (URL kommt dafür aus dem Submission-Daten, siehe weiter unten `call_back_url`). (Link)

Fachdienst/in Rolle	*Client*	*Server*

AS	Lieferung LP+AN-Submission-ID Daten an die VST	Hält pseudoynmisierte Daten inkl. AN + Submission-ID vor für das FDZ
VST	Liefert PüP+AN-Submission-ID ans FDZ	Empfänge PüP+AN-Submission-ID
FDZ	Bezieht pseudonmyisierte Daten inkl. AN + Submission-ID vom AS	Empfäng PüP+AN-Submission-ID

Übersetzungstabelle LP-AN+Submission-ID

Das AS sendet an die VST nach beidseitiger Authentisierung eine Übersetzungstabelle LP-AN+Submission-ID.

Übersetzungstabelle PüP-AN+Submission-ID

Das VST sendet an das FDZ eine Übersetzungstabelle PüP-AN+Submission-ID.

Das ist eine mit der Nomenklatur `Datenausleitung-VST-FDZ-<Submission-ID-Hex>.cbor`

Darin befindet sich folgende Datenstruktur.

```
Submission = [
   {"type": "vst->fdz",
    "created_at": Unix-Zeit,
    "submission_id": Submission-ID,
    "call_back_url": url_aktensystem},
    [PüP1, AN1], [PüP2, AN2] ...
]
```

Die Datenstruktur wird über CBOR binär kodiert.

Das FDZ erhält also bspw. `Datenausleitung-VST-FDZ-81b1b83e2736d45db16674e527890b8adc69ff7275acf11925faf44ae57f7e2f.cbor`

In der Datenstruktur selbst (CBOR-kodiert) sind die Submission-ID, PüP und AN binär kodiert (also jeweils 32 Byte Binärdaten).

Submission vom AS

Aus der `call_back_url` weiß das FDZ, welches AS es mit welcher URL kontaktieren muss. Das FDZ bezieht nach beidseitiger Authentisierung (siehe spätere Abschnitte) die Daten.

Die Daten sind "New-Line-Delimited-JSON (NDJSON)"-Dateien (mit bspw. folgenden Namen: `Datenausleitung-AS-FDZ-<Submission-ID-Hex>.ndjson`).

Vergleiche auch I_Data_Submission_Service.

Datensicherung

Wie in TLS+VAU-Protokoll beschrieben werden die Datentransporte jeweils über zwei übereinander liegende Sicherungsschichten (TLS+VAU-Protokoll) Vertraulichkeits- und Authentitätsgeschützt. Auf TLS-Ebene (HTTPS-Schnittstelle) gibt es nur eine Server-seitige Authentisierung. Auf der darüberliegen VAU-Protokoll-Ebene gibt es eine beidseitige Authentisierung (also sowohl VAU-Protokoll-Server-Authentisierung als auch VAU-Protokoll-Client-Authentisierung) -- Standardvorgehen bei ePA weil meistens ePA-Clients keine X.509-Zertifikate über die "direkte" Authentisierung bei TLS einsetzen können. Die nur Server-seitige TLS-Authentisierung bei der VST erleichtert die Umsetzung (spezielle Einsatzumgebung dort / Perimeter TLS-Gateway).

Lieferung LP+AN+Submission-ID durch die beiden AS an die VST

Wie in I_VST definiert hat die VST innerhalb der TI eine HTTPS-Schnittstelle und damit auch ein TLS-Serverzertifikat aus der TI-PKI (Komponenten-PKI der TI). Im TLS-Serverzertifikat gibt es eine OID oid_epa_vst als Profession-OID.

Auf Anwendungsebene erfolgt dann ein VAU-Protokoll-Verbindungsaufbau bei dem die VST ein C.FD.AUT-Zertifikat (TI-PKI) besitzt mit professionOID gleich oid_epa_vst.

Das AS als VAU-Protokoll-Client authentisiert sind wie üblich über PKI-basierte VAU-Protokoll-Client-Authentisierung beim VAU-Protokoll Verbindungsbaufbau. Zitat aus I_VST:

> "Erst nachdem sich das ePA-AS authentisiert hat und die VST die Authentifizierung erfolgreich durchführen konnte, darf das VST fachliche Zugriffe erlauben (Submission entgegennehmen)."

Ein AS stellt eine Submission her -- erzeugt dabei zufällig eine Submission-ID (256-Bit-Wert).

Innerhalb des VAU-Protokoll schickt das AS (= hier Client) in einem inneren Request per POST an `/epa/submission/api/v1/submissions/<Submission-ID-Hex>` die Übersetzungstabelle mit folgender Struktur (CBOR-kodiert):

```
Submission = [
   {"type": "as->vst",
    "created_at": Unix-Zeit,
    "submission_id": Submission-ID,
    "call_back_url": url_aktensystem},
    [LP1, AN1],
    [LP2, AN2], ...
]
```

Submission-ID ist ein 32 Byte Binärwert. Die Auftragsnummern (ANy) sind jeweils 32 Byte Binärwerte. Die Lieferpseudonyme (LPx) (Erzeugungsvorschrift wird ist in I_VST definiert) sind jeweils 103 Byte lange Binärwerte.

Die `call_back_url` verwendet das FDZ dann folgend um die pseduoynmisierten Daten vom Aktensystem per GET (TLS+VAU-Protokoll+Client-Authentisierung-FDZ) zu beziehen.

Die `call_back_url` muss mit `https://epa-as-<ePA-Anbieter-Zahl>.<Umgebung>.epa4all.de` beginnen. ePA-Anbieter-Zahl kann aktuell genau nur 1 oder 2 sein. Die "Umgebung" hat in der PU den Wert "prod".

In der `call_back_url` ist die Submission-ID schon vorhanden.

Lieferung von PüP+AN+Submission-ID vom VST an das FDZ

Für den Datentransfer der Übersetzungstabellen PüP+AN+Submission-ID vom VST an das FDZ hat die gematik keine Regelungshoheit. Die gematik schlägt vor die Schnittstelle genau so wie bei der VST (I_VST) und beim ePA-AS zu gestalten. Dann wäre die Schnittstellen alle beteiligten Kommunikationspartner gleich.

Dies wird im folgenden beschrieben.

Das FDZ besitzt eine HTTPS-Schnittstelle (TLS-Server-Zertifikat aus der TI-PKI, im Zertifikat oid_epa_fdz). Auf Anwendungsebene besitzt das FDZ ein C.FD.AUT-Zertifikat (TI-PKI) mit der OID oid_epa_fdz.

Kommunikationspartner	*Zertifikate*
ePA-AS	TLS-Server-Zertifikat mit oid_epa_dvw C.FD.AUT-Zertifikat mit oid_epa_vau
VST	TLS-Server-Zertifikat mit oid_epa_vst C.FD.AUT-Zertifikat mit oid_epa_vst
FDZ	TLS-Server-Zertifikat mit oid_epa_fdz C.FD.AUT-Zertifikat mit oid_epa_fdz

Die VST baut auf Anwendungsebene eine VAU-Protokoll-Verbindung auf. Darin authentisiert sich die VST über PKI-basierte VAU-Protokoll-Client-Authentisierung, genau wie zuvor das AS gebenüber der VST und später das FDZ gegenüber dem AS.

Anschließend sendet dei VST über einen innere HTTP-Request per POST an `/epa/submission/api/v1/submissions/<Submission-ID-Hex>` eine Abbildungstabelle PüP+AN+Submission-ID wie in I_VST definiert.

```
Submission = [
   {"type": "vst->fdz",
```

```
    "created_at": Unix-Zeit,
    "submission_id": Submission-ID,
    "call_back_url": url_aktensystem},
    [PüP1, AN1],
    [PüP2, AN2], ...
]
```

Submission-ID ist ein 32 Byte Binärwert. Die Auftragsnummern (ANx) sind ebenfalls 32 Byte Binärwerte. Die PüP sind Binärwerte.

Die `call_back_url` muss mit `https://epa-as-<ePA-Anbieter-Zahl>.<Umgebung>.epa4all.de` beginnen. ePA-Anbieter-Zahl kann aktuell genau nur 1 oder 2 sein. Die "Umgebung" hat in der PU den Wert "prod".

In der `call_back_url` ist die Submission-ID schon vorhanden.

Abholung von Submissions durch das FDZ von den AS

Das FDZ tritt hier als "normaler" ePA-Client auf. Es verbindet sich per TLS an den in der `call_back_url` angegeben Host (FQDN in der URL).

Die `call_back_url` muss mit `https://epa-as-<ePA-Anbieter-Zahl>.<Umgebung>.epa4all.de` beginnen. ePA-Anbieter-Zahl kann aktuell genau nur 1 oder 2 sein. Die "Umgebung" hat in der PU den Wert "prod".

Anschließend baut es über das VAU-Protokoll (zu verwendenen Pfade dort definiert) eine VAU-Protokoll-Verbindung auf. Anschließend authentisiert sich das FDZ über die PKI-basierte VAU-Protokoll-Client-Authentisierung.

Nun kann es per GET im inneren Request per GET die im `call_back_url` aufgeführte Submission abholen.

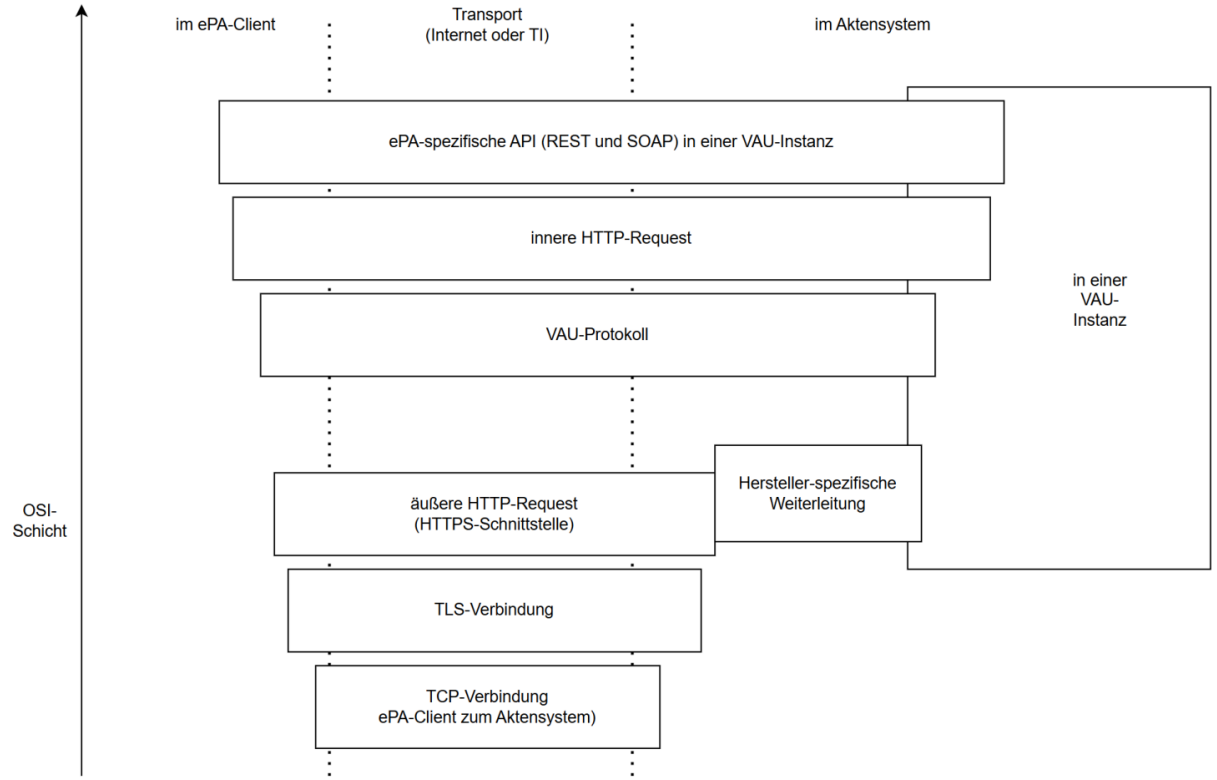
Beispiel für `call_back_url`:

> https://epa-as-1.prod.epa4all.de/epa/submission/api/v1/submissions/6f3e4ad45a4c1c363f3e9b797ecca58c4f72998f83defe0b3fe867a988c9b362

Vergleiche auch I_Data_Submission_Service.

TLS+VAU-Protokoll

Bei ePA wird bei den Kommunikationen zwei Sicherungsschichten (TLS+VAU-Protokoll) parallel verwendet.



Zunächst gibt es eine TLS-Schicht, die die Grundlage einer HTTPS-Schnittstelle ist. Auf der TLS-Ebene gibt es ein TLS-Server-Zertifikat aus der TI-PKI.

Über diese HTTPS-Schnittstelle werden "äußere HTTP-Request" vom Client an den Server (VST, FDZ, ePA-AS) gesendet.

In diesen äußeren Requests sind VAU-Protokoll-Nachrichten verpackt. Eine zentrale Grundlage des VAU-Protokolls stellt das VAU-Protokoll-Server-Zertifikat -- wieder aus der Komponenten-PKI der TI.

VAU-Protokoll:

Spezifikation:	https://gemspec.gematik.de/docs/gemSpec/gemSpec_Krypt/latest/#7
Beispiel-Code für Aushandlung der symmetrischen VAU-Kanal-Schlüssel:	https://bitbucket.org/andreas_hallof/vau-protokoll/src/master/minimal/

Beispiel-Code in python	https://bitbucket.org/andreas_hallof/vau-protokoll/src/master/
Beispiel-Code für Java:	https://github.com/gematik/lib-vau
Beispiel-Code in C#:	https://github.com/gematik/lib-vau-csharp
Beispiel-Code in Go:	https://github.com/gematik/zero-lab/tree/vau/pkg/libvau