IRD Technische Spezifikation für Versicherungsträger (API-Version 1.0)

Dokumenteninformation

Version	Datum	Grund der Änderung, besondere Hinweise	Bearbeiter
1.0	18.04.2024	Initialversion	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.1	07.05.2024	- Korrektur Angaben zum Vitalstatus (Enum) - Änderungen bzgl. nur Rückgabe von Fehlern - Ergänzung Authentisierung und Authentifizierung - redaktionelle Änderungen	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.2	23.05.2024		RKI Referat VIG (Vertrauensstelle-ird@rki.de)

Inhaltsverzeichnis

Einleitung

Diese technische Spezifikation beschreibt die Kommunikation eines meldenden Versicherungsträgers über die REST-Schnittstelle der Vertrauensstelle IRD für die Meldungen entsprechend §17 Abs. 2 IRegG und § 18 Abs. 1 IRegBV. Das sind die halbjährlichen und anlassbezogenen Meldungen der Vitalstatus der im Implantateregister Deutschland erfassten Versicherten sowie die Information zum Wechsel des Versicherungsträgers dieses Personenkreises. Als Versicherungsträger sind hier die gesetzlichen Krankenkassen, die privaten Krankenversicherungsunternehmen und die sonstigen Kostenträger nach §17 Abs. 3 IRegG zusammengefasst.

Inhaltliche Pflichtangaben sind Felder, die sich mindestens aus §17 Abs. 2 IRegG (siehe https://www.gesetze-im-internet.de/iregg/__17.html) begründen sowie weitere technische Felder.

Allgemeine Informationen

Die API Version 1.0 akzeptiert als patienten-identifizierendes Datum den unveränderlichen Teil der Krankenversichertennummer (KVNR) nach § 290 Absatz 1 Satz 2 SGB V oder eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer nach § 17 Absatz 4 Satz 3 IRegG[1]. Aktuell befindet sich die Spezifikation für diese Identifikationsnummer noch in Abstimmung.

Umgebungen

Die Vertrauensstelle ist innerhalb der Referenzumgebung (RU) der Telematikinfrastruktur unter https://vst-ird-ru.rki-ti.de zu erreichen und innerhalb der Produktivumgebung (PU) der Telematikinfrastruktur unter https://vst-ird.rki-ti.de.

Registrierung an der Schnittstelle

Zur Verwendung der Schnittstelle ist eine einmalige Registrierung notwendig. Dazu wird der Vertrauensstelle auf einem geeigneten, noch festzulegenden Weg ein signiertes Paket mit den Angaben der Telematik-Id sowie der Haupt-IK (Haupt-Institutionskennzeichen) des Versicherungsträgers zugesendet. Die Vertrauensstelle prüft dann die Signatur und nimmt die Registrierung des Versicherungsträgers zur Benutzung dieses Services auf.

Die Signierung von Daten mit dem Schlüsselmaterial der TI ist nur mithilfe des Konnektors bzw. des Basis-/KTR-Consumers der TI möglich. Die Signierung der Daten für die Registrierung an der Schnittstelle wird über die SOAP-Aktion "ExternalAuthenticate" mit dem AUT-Zertifikat der SMC-B realisiert

Als Daten, die durch die SOAP-Action signiert werden sollen, werden die Bytes der beiden Zeichenfolgen Telematik-Id und Haupt-IK, konkateniert durch das Zeichen "|", gesendet.
Ein Beispiel dazu:

```
Telematik-Id: 3-17.2.1234567000.10.432
Haupt-IK: 123456789
<Wert der Telematik-Id> + | + <Wert der Haupt-IK>

Hexadezimale Darstellung:
33 2D 31 37 2E 32 2E 31 32 33 34 35 36 37 30 30 30 2E 31 30 2E 34 33 32 7C 31 32 33 34 35 36 37 38 39

Base64-Darstellung:
My0xNy4yLjEyMzQ1NjcwMDAuMTAuNDMyfDEyMzQ1Njc4OQ==
```

(Test-Telematik-Id wurde von dieser Seite genommen: https://www.akwl.de/news.php?nid=606)

Das signierte Paket muss dann mit der Angabe des verwendeten AUT-Zertifikats sowie den Angaben der Telematik-Id und der Haupt-IK des Versicherungsträgers auf einem geeigneten, noch festzulegenden Weg mitgeteilt werden.

Weitere Informationen zur Signaturerstellung mit der SOAP-Aktion "ExternalAuthenticate" sowie dem Abruf des AUT-Zertifikates finden sich in den übergreifenden Beispielen.

Datenformat

Eingabeformat

Die Daten müssen in dem Format **application/json** zu der Schnittstelle gesendet werden. Die Zeichenkodierung wird als **UTF-8** gelesen. Namen der Eigenschaften werden CaseInsensitive ausgewertet.

Festlegungen zu den JSON-Daten

- es werden keine Kommentare in den JSON-Daten akzeptiert. Die Verwendung von Kommentaren führt zu einem Fehler.
- es werden keine doppelten Schlüssel (also Namen der Eigenschaften) in einem JSON-Knoten akzeptiert. Die Verwendung von doppelten Schlüsseln führt zu einem Fehler.
- es werden nur Eigenschaften akzeptiert, die auch in der OpenAPI-Definition definiert sind. Die Verwendung anderer, uns unbekannter Eigenschaften führt zu einem Fehler.

Rückgabe

Nach erfolgreicher Annahme und rudimentärer Prüfung der Daten wird der HTTP-Statusode 200 zurückgegeben. Der (Content) der Antwort ist dabei leer.

Authentisierung und Authentifizierung

Das Verfahren für Authentisierung bzw. Authentifizierung und Autorisierung wird mithilfe eines signierten Anmeldetokens durchgeführt. Dazu wird mithilfe des Signatur-Zertifikats der SMC-B-Karte eine Signatur über die Werte Telematik-Id und Haupt-IK des Versicherungsträgers gebildet. Mithilfe dieses Tokens, das als Authentication-Header mit den zu übertragenden Daten mitgesendet wird, kann die Vertrauensstelle diese Anmeldung prüfen.

Das Verfahren sowie die konkreten Formate, die benutzt werden müssen, sind aktuell in Klärung.

Signierung von Daten

Signierung der Eingangsdaten

Signaturen der Daten, die von den Versicherungsträgern mit den jeweiligen Daten mitzusenden sind, gewährleisten die Integrität und die Authentizität der übertragenenen Daten. Es wird hier das Schema Encrypt-Then-Sign angewendet, die Signatur wird also über die bereits verschlüsselten Daten gebildet.

Bei der Signatur handelt es sich um eine nonQES - Signatur (nicht qualifizierte elektronische Signatur).

Bei der Signatur werden die Eingangsdaten in einer konkatenierten Form zur Signaturberechnung und -prüfung betrachtet.

Für beispielsweise einem JSON-Fragment mit diesen Daten


```
{
  "IdDatenlieferung": "2024-01",
  "Meldungen": [
    {
      "IdDatensatz": "2024-01-001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBkYXMgdmVyc2NobMO8c3NlbHRlIFN0ZXJiZWRhdHVt"
    },
    {
      "IdDatensatz": "2024-01-002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVzIFN0ZXJiZWRhdHVt"
    }
  ]
}
```


entsteht folgendes Ergebnis:

```
<Wert der Eigenschaft IdDatenlieferung> + | + <Wert des IdDatensatzes Datensatz 1> + | + <Wert des
IdVersicherten Datensatz 1> + | + <Wert des VitalStatus Datensatz 1> + | + <Wert des Sterbedatum Datensatz 1> +
| + <Wert des IdDatensatzes Datensatz 2> + | + <Wert des IdVersicherten Datensatz 2> + | + <Wert des
Vitalstatus Datensatz 2> + | + <Wert des Sterbedatum Datensatz 2>

Hexadezimale Darstellung (anhand des Beispiels oben)
32 30 32 34 2D 30 31 7C 32 30 32 34 2D 30 31 2D 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63
68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 7C 49
63 68 20 62 69 6E 20 64 65 72 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 20 56 69 74 61 6C 73 74 61 74 75
73 7C 63 68 20 62 69 6E 20 64 61 73 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 20 53 74 65 72 62 65 64 61
74 75 6D 7C 32 30 32 34 2D 30 31 2D 30 30 32 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76
65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74
65 6E 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74
65 72 20 57 65 72 74 20 64 65 73 20 56 69 74 61 6C 73 74 61 74 75 73 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61
6E 64 65 72 65 73 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 73 20 53 74 65 72 62 65 64 61 74 75 6D
```

Die Plain Bytes werden an die entsprechende Schnittstelle des Konnektors bzw. des Basis-/KTR-Consumer gesendet. Nach der Operation der Signaturbildung wird dann die Signatur zurückgegeben. Diese ist dann in der Eigenschaft "Signatur" zu verwenden. Erweiterte Information zur Verwendung der Schnittstelle des Konnektors bzw. des Basis-/KTR-Consumer zur Signaturbildung und zu den konkreten Formaten finden sich unter dem Punkt Übergreifende Beispiele.

Das jeweilige Beispiel zur Bildung der Zeichenkette, die zur Signaturbildung verwendet werden muss, findet sich unter den jeweiligen API.

Signierung der Ausgangsdaten

Signaturen der Daten, die von der Vertrauensstelle mit der jeweiligen Antwort mitgesendet werden, gewährleisten die Integrität und die Authentizität der übertragenenen Daten.

Bei der Signatur handelt es sich um eine ECDSA-Signatur, die über den SHA256-Hash der Daten der Antwort gebildet wird.

Erweiterte Information zur Prüfung der Signatur und zu den konkreten Formaten finden sich unter dem Punkt Übergreifende Beispiele.

> Die von der Vertrauensstelle ausgestellte Signatur ist vor der Verarbeitung oder Weiterleitung von Daten an die Registerstelle zu prüfen.

Der öffentliche Schlüssel der Vertrauensstelle IRD für die Prüfung von Signaturen wird innerhalb der TI für die Versicherungsträger an einem gesonderten Endpunkt zum regelmäßigen Download bereitgestellt.

Verschlüsselung von Daten

Verschlüsselung der Eingangsdaten

Der Schutz der Datenübertragung (Transportkanal) durch die Versicherungsträger an die Vertrauensstelle IRD muss per TLS gesichert sein. Technisch bedingt sind für eine durchgehende Ende-zu-Ende-Verschlüsselung alle patienten- und/oder fallidentifizierenden Daten vor der Übertragung für die Vertrauensstelle auf Feldebene zu verschlüsseln.

Erweiterte Information zur konkreten Umsetzung finden sich unter dem Punkt Übergreifende Beispiele.

Die öffentlichen Schlüssel der Vertrauensstelle IRD und der Registerstelle IRD für die Verschlüsselung werden innerhalb der TI für die Versicherungsträger an gesonderten Endpunkten zum regelmäßigen Download bereitgestellt.

HTTP-Programmierschnittstellen (APIs)

> **Allgemeiner Hinweis**
>
> ⚠ Innerhalb der Eigenschaften "IdDatenlieferung" und "IdDatensatz" dürfen keine patienten- bzw. fallidentifizierende Daten übermittelt werden

API für die Übertragung von Vitalstatusmeldungen für im Implantateregister erfasste Versicherte

POST /notify/api/v1/vitalstatusnotification

Diese Schnittstelle ist für die Übertragung von Vitalstatusmeldungen für im Implantateregister erfasste Versicherte ("Implantatträger") zu verwenden.

Eingabedaten:

Die Daten, die von der API für die Übertragung von Vitalstatusmeldungen von im Implantateregister erfassten Versicherten erwartet und verarbeitet werden, sind in der OpenAPI-Spezifikation OpenAPI_VitalStatusNotificationService-API_V1.json unter dem Typ "VitalStatusNotificationBundle" beschrieben. Die Dokumente zum API werden zukünftig unterhalb der URL https://xml.ir-d.de/rst/download/vst/ zum Download bereitstehen.

Eigenschaft "IdDatenlieferung"

- definiert die für den meldenden Versicherungsträger eindeutige Id der Datenlieferung
- diese wird durch den Versicherungsträger festgelegt
- Optional: Nein
- Typ: String
- Schutz: Der Wert wird in Plaintext übertragen

Eigenschaft "Meldungen"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "VitalStatusNotification"
- Felder innerhalb eines Datensatzes:
    - IdDatensatz
        - definiert die eindeutige Id des Datensatzes innerhalb dieser Datenlieferung
        - diese wird durch den Versicherungsträger festgelegt
        - Optional: Nein
        - Typ: String
        - Schutz: Der Wert wird in Plaintext übertragen
    - IdVersicherter
        - definiert den unveränderlichen und eindeutigen Identifikator eines Versicherten. Dieser Identifikator entspricht in der Regel dem unveränderbarem Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Alternativ besteht die Möglichkeit für eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Als personenidentifizierendes Merkmal muss dieser Wert für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - bei Angabe einer Krankenversichertennummer:
                - es wird geprüft, dass die Angabe einer gültigen KVNR entspricht (Regex: ^[A-Z]{1}[0-9]{9}$, sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass die Angabe nicht der für Patienten ohne deutsche Krankenversicherung reservierten KVNR entspricht ([1])
                - es wird geprüft, dass diese KVNR im jeweiligen Kontext benutzt werden darf ([2])
            - bei Angabe einer anderen eindeutigen, unveränderbaren und nach einheitlichen Kriterien gebildeten Identifikationsnummer:
                - es wird geprüft, ob die Angabe einem bekannten, festgelegtem Schema entspricht, einschließlich der Prüfung einer Prüfsumme
    - Sterbedatum
        - Sterbedatum des Patienten (unverschlüsselte Form: yyyy-mm-dd)
        - Optional: Nein. Wenn diese Angabe nicht anwendbar ist (da der VitalStatus des Versicherten "Lebend" oder "Unbekannt" ist, so muss dieses Feld durch den Ersatzwert "---N/A----" anstelle des Todesdatums befüllt werden (als Klartext vor der Verschlüsselung). Dieses Feld wird immer genutzt, damit durch fehlendes bzw. gesetztes Feld nicht indirekt auf den Vitalstatus des Eintrags geschlossen werden kann.
        - Format: Byte
        - Typ: String
        - Schutz: Als medizinisches Datum muss dieser Wert für die Registerstelle verschlüsselt werden
        - Prüfungen (diese werden asynchron bei der Registerstelle nach der Entschlüsselung vorgenommen)
            - Die Registerstelle prüft, ob die Angabe einem gültigen Datum mit erwartetem Format yyyy-mm-dd oder dem definierten Ersatzwert "---N/A----" entspricht.
    - Vitalstatus
        - aktueller Vitalstatus des Versicherten
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Als medizinisches Datum muss dieser Wert für die Registerstelle verschlüsselt werden
        - Prüfungen (diese werden asynchron bei der Registerstelle nach der Entschlüsselung vorgenommen)
            - es wird geprüft, ob die vorhandene Angabe einem der folgenden erlaubten Werte entspricht: "01"=Lebend, "02" =Verstorben, "03"=Unbekannt

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Rückgabe

Nach erfolgreicher Annahme und rudimentärer Prüfung der Daten wird der HTTP-Statusode 200 zurückgegeben. Der (Content) der Antwort ist dabei leer.

Fehlerfälle

Fehler, die bei der Annahme oder Prüfung eines Datenpaketes an das API der Vertrauensstelle IRD auftreten, werden auf die gesamte Meldung ausgewertet. Auftretende Fehler werden dem meldenden Versicherungsträger sofort synchron zurückgemeldet.

Weiterhin erfolgt nach der Annahme der Daten innerhalb der Vertrauensstelle und im Implantateregister eine Verarbeitung der Daten. Auftretende Fehler bei der Verarbeitung werden dem meldenden Versicherungsträger asynchron über einen gesonderten Abruf der Daten zur Verfügung gestellt. Die Abfrage von Fehlern wird durch ein Polling-Mechanismus des meldenden Versicherungsträgers erreicht. Wenn zu Datensätzen keine Fehlermeldungen zum Abruf zur Verfügung stehen, so bedeutet es für den Versicherungsträger, dass keine Fehlrer bei der Verarbeitung aufgetreten sind.

Synchrone Fehlermeldungen

Bei Fehlern, die sich auf die gesamte Meldung beziehen, findet **keine** Verarbeitung der Datensätze statt.

Fehler, die bei der Kommunikation mit der API oder bei der Verarbeitung von Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 400: dieser Fehler wird zurückgemeldet, wenn die Nachricht fehlerhaft war, weil Pflichtangaben fehlen, oder Angaben nicht plausibel sind (siehe dazu Prüfungen von Feldern). Es werden in diesem Fehlerfall keine erweiterten Fehlerbeschreibungen zurückgemeldet.
- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war. Dies beinhaltet auch die Verwendung von produktiven Daten in der Referenz-Umgebung.
- StatusCode 415: dieser Fehler wird zurückgemeldet, wenn ein falscher Content-Type gesendet wurde.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Request


```
{
  "IdDatenlieferung": "2024-01",
  "Meldungen": [
    {
      "IdDatensatz": "2024-01-001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBkYXMgdmVyc2NobMO8c3NlbHRlIFN0ZXJiZWRhdHVt"
    },
    {
      "IdDatensatz": "2024-01-002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVzIFN0ZXJiZWRhdHVt"
    }
  ],
  "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```


Signaturbildung der Eingangsdaten

Die Signatur, die vom Sender an den Eingangsdaten mitzusenden ist, bezieht sich auf die (bereits verschlüsselten) Vitalstatus-Meldungen. Bei der Signatur werden die Daten in einer konkatenierten Form zur Signaturberechnung und -prüfung betrachtet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
  "IdDatenlieferung": "2024-01",
  "Meldungen": [
    {
      "IdDatensatz": "2024-01-001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBkYXMgdmVyc2NobMO8c3NlbHRlIFN0ZXJiZWRhdHVt"
    },
    {
      "IdDatensatz": "2024-01-002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Sterbedatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVyIFN0ZXJiZWRhdHVt"
    }
  ]
}
```


entsteht folgendes Ergebnis:

```
<Wert der Eigenschaft IdDatenlieferung> + | + <Wert des IdDatensatzes Datensatz 1> + | + <Wert des
IdVersicherten Datensatz 1> + | + <Wert des VitalStatus Datensatz 1> + | + <Wert des Sterbedatum Datensatz 1> +
| + <Wert des IdDatensatzes Datensatz 2> + | + <Wert des IdVersicherten Datensatz 2> + | + <Wert des
Vitalstatus Datensatz 2> + | + <Wert des Sterbedatum Datensatz 2>

Hexadezimale Darstellung
32 30 32 34 2D 30 31 7C 32 30 32 34 2D 30 31 2D 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63
68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 7C 49
63 68 20 62 69 6E 20 64 65 72 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 20 56 69 74 61 6C 73 74 61 74 75
73 7C 63 68 20 62 69 6E 20 64 61 73 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 20 53 74 65 72 62 65 64 61
74 75 6D 7C 32 30 32 34 2D 30 31 2D 30 30 32 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76
65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74
65 6E 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74
65 72 20 57 65 72 74 20 64 65 73 20 56 69 74 61 6C 73 74 61 74 75 73 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61
6E 64 65 72 65 73 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 73 20 53 74 65 72 62 65 64 61 74 75 6D
```

Weitere Informationen zu der Bildung der Signatur finden sich unter dem Punkt Signierung von Eingangsdaten.

API für den Abruf von Anfragen zu Vitalstatusmeldungen für im Implantateregister erfasste Versicherte

GET /notify/api/v1/vitalstatusnotification/requests

Diese Schnittstelle ist für den Abruf von Anfragen zu Vitalstatusmeldungen für im Implantateregister erfasste Versicherte ("Implantatträger") zu verwenden.

Eingabedaten:

keine

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der StatusCode 204 zurückgegegben.

Wenn Daten zum Abruf vorhanden sind, wird der StatusCode 200 zurückgegeben. Als Content der Antwort wird die die Liste der Versicherten zurückgegeben, zu denen eine Vitalstatusmeldung angefragt wird. Elemente innerhalb dieser Liste sind vom Typ "RequestsForVitalStatusNotificationBundle".

Eigenschaft "Meldungen"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "RequestForVitalStatusNotification"
- Felder innerhalb eines Datensatzes:
    - IdVersicherter
        - definiert den unveränderlichen und eindeutigen Identifikator eines Versicherten. Dieser Identifikator entspricht in der Regel dem unveränderbarem Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V).

Alternativ besteht die Möglichkeit für eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
- Optional: Nein
- Format: Byte
- Typ: String
- Schutz: Als personenidentifizierendes Merkmal ist dieser Wert für den Versicherungsträger verschlüsselt

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Fehlerfälle

Synchrone Fehlermeldungen

Fehler, die beim Abruf der Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response

```
{
        "Anfragen":
        [
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg=="
            },
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```

Signierung der Ausgangsdaten

Die Antwort, die durch die Aktion zurückgegeben werden, beinhalten neben der Liste der Daten auch die Signatur. Die Signatur wird dabei über die konkatenierten Werte innerhalb dieser Antwort gebildet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
        "Anfragen":
        [
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg=="
            },
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
              }
        ]
}
```

entsteht folgendes Ergebnis:

```
<Wert IdVersicherter1> + | + <Wert IdVersicherter2>

Hexadezimale Darstellung
49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73
20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76
65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74
65 6E
```

Weitere Informationen finden sich unter dem Punkt Signierung der Ausgangsdaten.

API für den Abruf der Benachrichtigungen über die Anonymisierung der Daten von im Implantateregister erfassten Versicherten

GET /notify/api/v1/vitalstatusnotification/anonymizationnotifications

Entsprechend IRegG benachrichtigt das IRD die Versicherungsträger bei der Anonymisierung der Daten zu einem im Implantateregister erfassten Versicherten ("Implantatträger"). Diese Schnittstelle ist für den Abruf dieser Benachrichtigungen zu verwenden.

Eingabedaten:

keine

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der StatusCode 204 zurückgegegben.

Wenn Daten zum Abruf vorhanden sind, wird der StatusCode 200 zurückgegeben. Als Content der Antwort wird die Liste der Versicherten zurückgegeben, zu denen eine Anonymisierung innerhalb der Vertrauensstelle und der Registerstelle vorgenommen wurden. Elemente innerhalb dieser Liste sind vom Typ "AnonymizationNotificationsBundle".

Eigenschaft "Anonymisierungen"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "AnonymizationNotification"
- Felder innerhalb eines Datensatzes:
    - IdVersicherter
        - definiert den unveränderlichen und eindeutigen Identifikator eines Versicherten. Dieser Identifikator entspricht in der Regel dem unveränderbarem Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Alternativ besteht die Möglichkeit für eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Als personenidentifizierendes Merkmal ist dieser Wert für den Versicherungsträger verschlüsselt

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Fehlerfälle

Synchrone Fehlermeldungen

Fehler, die beim Abruf der Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response

```
{
        "Anonymisierungen":
        [
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg=="
            },
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```

Signierung der Ausgangsdaten

Die Antwort, die durch die Aktion zurückgegeben werden, beinhalten neben der Liste der Daten auch die Signatur. Die Signatur wird dabei über die konkatenierten Werte innerhalb dieser Antwort gebildet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
        "Anonymisierungen":
        [
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg=="
            },
            {
                        "IdVersicherter":
"SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
              }
        ]
}
```

entsteht folgendes Ergebnis:

```
<Wert IdVersicherter1> + | + <Wert IdVersicherter2>

Hexadezimale Darstellung
49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73
20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 61 6E 64 65 72 65 72 20 76
65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74
65 6E
```

Weitere Informationen finden sich unter dem Punkt Signierung der Ausgangsdaten.

API für den Abruf von Fehlern, die bei der asynchronen Verarbeitung von Vitalstatus-Meldungen auf Seiten der Vertrauensstelle oder der Registerstelle aufgetreten sind

GET /notify/api/v1/vitalstatusnotification/processingerrors

Diese Schnittstelle ist für den Abruf von Fehlern, die bei der asynchronen Verarbeitung von Vitalstatus-Meldungen auf Seiten der Vertrauensstelle oder der Registerstelle aufgetreten sind.

Eingabedaten:

keine

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der StatusCode 204 zurückgegegben.

Wenn Daten zum Abruf vorhanden sind, wird der StatusCode 200 zurückgegeben. Als Content der Antwort wird die Liste der Datensätze zurückgegeben, bei denen es bei der asynchronen Verarbeitung von Vitalstatus-Meldungen auf Seiten der Vertrauensstelle oder der Registerstelle zu Fehlern kam. Elemente innerhalb dieser Liste sind vom Typ "VitalStatusNotificationProcessingErrorsBundle".

Eigenschaft "Fehler"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "VitalStatusNotificationProcessingErrorResult"
- Felder innerhalb eines Datensatzes:
  - IdDatenlieferung
    - definiert die für den meldenden Versicherungsträger eindeutige Id der Datenlieferung des Datensatzes
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen
  - IdDatensatz
    - definiert die eindeutige Id des Datensatzes (innerhalb der Datenlieferung)
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen
  - Code
    - definiert den Fehler, der aufgetreten ist, und wodurch der Datensatz nicht verarbeitet werden konnte
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Fehlerfälle

Synchrone Fehlermeldungen

Fehler, die beim Abruf der Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response


```
{
        "Fehler":
        [
            {
                "IdDatenlieferung": "2024-01",
                        "IdDatensatz": "2024-01-001",
                        "Code": "DecryptionError"
            },
            {
            "IdDatenlieferung": "2024-02",
                        "IdDatensatz": "2024-02-002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```


Signierung der Ausgangsdaten

Die Antwort, die durch die Aktion zurückgegeben werden, beinhalten neben der Liste der Daten auch die Signatur. Die Signatur wird dabei über die konkatenierten Werte innerhalb dieser Antwort gebildet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
        "Fehler":
        [
            {
                "IdDatenlieferung": "2024-01",
                        "IdDatensatz": "2024-01-001",
                        "Code": "DecryptionError"
            },
            {
            "IdDatenlieferung": "2024-02",
                        "IdDatensatz": "2024-02-002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ]
}
```

entsteht folgendes Ergebnis:

```
<Wert IdDatenliegferung1> + | + <Wert IdDatensatz1> + | + <Wert Fehlercode1> + <Wert IdDatenliegferung2> + | +
<Wert IdDatensatz2> + | + <Wert Fehlercode2>

Hexadezimale Darstellung
32 30 32 34 2D 30 31 7C 32 30 32 34 2D 30 31 2D 30 30 31 7C 44 65 63 72 79 70 74 69 6F 6E 45 72 72 6F 72 7C 32
30 32 34 2D 30 32 7C 32 30 32 34 2D 30 32 2D 30 30 32 7C 57 72 6F 6E 67 46 6F 72 6D 61 74 49 64 56 65 72 73 69
63 68 65 72 74 65 72
```

Weitere Informationen finden sich unter dem Punkt Signierung der Ausgangsdaten.

API für die Übertragung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten

POST /notify/api/v1/insuranceupdatenotification

Diese Schnittstelle ist für die Übertragung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten zu verwenden.

Eingabedaten:

Die Daten, die von der API für die Übertragung von Vitalstatusmeldungen von im Implantateregister erfassten Versicherten erwartet und verarbeitet werden, sind in der OpenApi-Spezifikation unter der Adresse https://xml.ir-d.de/rst/download/vst/OpenAPI_InsuranceUpdateService-API_V1.json unter dem Typ "VitalStatusNotificationBundle" beschrieben. Die Dokumente zum API werden zukünftig unterhalb der URL https://xml.ir-d.de/rst/download/vst/ zum Download bereitstehen.

Eigenschaft "IdDatenlieferung"

- definiert die für den meldenden Versicherungsträger eindeutige Id der Datenlieferung
- diese wird durch den Versicherungsträger festgelegt
- Optional: Nein
- Typ: String
- Schutz: Der Wert wird in Plaintext übertragen

Eigenschaft "Meldungen"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "InsuranceUpdateNotification"
- Felder innerhalb eines Datensatzes:
    - IdDatensatz
        - definiert die eindeutige Id des Datensatzes innerhalb dieser Datenlieferung
        - diese wird durch den Versicherungsträger festgelegt
        - Optional: Nein
        - Typ: String
        - Schutz: Der Wert wird in Plaintext übertragen
    - IdVersicherter
        - definiert den eindeutigen Identifizierer des Versicherten. Dieser Identifizierer ist der unveränderbare Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V) bzw. eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
        - Optional: Nein

        - Format: Byte
        - Typ: String
        - Schutz: Der Wert muss für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - bei Angabe einer Krankenversichertennummer:
                - es wird geprüft, dass die Angabe einer gültigen KVNR entspricht (Regex: ^[A-Z]{1}[0-9]{9}$, sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass die Angabe nicht der für Patienten ohne deutsche Krankenversicherung reservierten KVNR entspricht ([1])
                - es wird geprüft, dass diese KVNR im jeweiligen Context benutzt werden darf ([2])
            - bei Angabe einer anderen eindeutigen, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer:
                - es wird geprüft, ob die Angabe einen bekannten, festglegten Schema entspricht, sowie eine Prüfung der Prüfsumme
    - IdVersicherterAlt
        - definiert den eindeutigen Identifizierer des Versicherten bei der vorherigen Versicherung. Dieser Identifizierer ist der unveränderbare Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V) bzw. eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
        - Optional: Ja - Muss nur gesetzt werden, wenn der Versicherte bei seiner Vorversicherung unter einer anderen Nummer versichert war
        - Format: Byte
        - Typ: String
        - Schutz: Der Wert muss für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - bei Angabe einer Krankenversichertennummer:
                - es wird geprüft, dass die Angabe einer gültigen KVNR entspricht (Regex: ^[A-Z]{1}[0-9]{9}$, sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass die Angabe nicht die der definierten KVNR eines Patienten ohne deutsche Krankenversicherung ist ([1])
                - es wird geprüft, dass diese KVNR im jeweiligen Context benutzt werden darf ([2])
            - bei Angabe einer anderen eindeutigen, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer:
                - es wird geprüft, ob die Angabe einen bekannten, festglegten Schema entspricht, sowie eine Prüfung der Prüfsumme
    - IkAlt
        - definiert die Haupt-IK (Institutionskennzeichen) der Vorversicherung
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Der Wert muss für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - es wird geprüft, dass die Angabe die eines validen Institutionskennzeichen entspricht (Regex: ^[0-9]{9}$, sowie Prüfung der Prüfsumme)

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Rückgabe

Nach erfolgreicher Verarbeitung der Daten wird der StatusCode 200 zurückgegeben. Als Content der Antwort wird die signierte Transfernummer zurückgegegben (Typ: "SignedTransferNumber").

Fehlerfälle

Fehler, die bei der Annahme oder Prüfung eines Datenpaketes an das API der Vertrauensstelle IRD auftreten, werden auf die gesamte Meldung ausgewertet. Auftretende Fehler werden dem meldenden Versicherungsträger sofort synchron zurückgemeldet.

Weiterhin erfolgt nach der Annahme der Daten innerhalb der Vertrauensstelle eine Verarbeitung der Daten. Auftretende Fehler bei der Verarbeitung werden dem meldenden Versicherungsträger asynchron über einen gesonderten Abruf der Daten zur Verfügung gestellt. Die Abfrage von Fehlern wird durch ein Polling-Mechanismus des meldenden Versicherungsträgers erreicht. Wenn zu Datensätzen keine Fehlermeldungen zum Abruf zur Verfügung stehen, so bedeutet es für den Versicherungsträger, dass keine Fehlrer bei der Verarbeitung aufgetreten sind.

Fehler, die bei der Kommunikation mit der API oder bei der Verarbeitung von Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 400: dieser Fehler wird zurückgemeldet, wenn die Nachricht fehlerhaft war, weil Pflichtangaben fehlen, oder Angaben nicht plausibel sind (siehe dazu Prüfungen von Feldern). Es werden in diesem Fehlerfall keine erweiterten Fehlerbeschreibungen zurückgemeldet.
- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war. Dies beinhaltet auch die Verwendung von produktiven Daten in der Referenz-Umgebung.
- StatusCode 415: dieser Fehler wird zurückgemeldet, wenn ein falscher Content-Type gesendet wurde.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Request

```
{
  "IdDatenlieferung": "2024-001",
  "Meldungen": [
    {
      "IdDatensatz": "2024-01-001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "IkAlt": "SWNoIGJpbiBlaW4gdmVyc2ljaGVydGVyIFdlcnQgZGVyIElrQWx0"
    },
    {
      "IdDatensatz": "2024-01-002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVy",
      "IdVersicherterAlt":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVyQWx0",
      "IkAlt": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElrQWx0"
    }
  ],
  "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```

Signaturbildung der Eingangsdaten

Die Signatur, die vom Sender an den Eingangsdaten mitzusenden ist, bezieht sich auf die (bereits verschlüsselten) Meldungen zu den Wechsel der Versicherungen. Bei der Signatur werden die Daten in einer konkatenierten Form zur Signaturberechnung und -prüfung betrachtet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
  "IdDatenlieferung": "2024-001",
  "Meldungen": [
    {
      "IdDatensatz": "2024-01-001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "IkAlt": "SWNoIGJpbiBlaW4gdmVyc2ljaGVydGVyIFdlcnQgZGVyIElrQWx0"
    },
    {
      "IdDatensatz": "2024-01-002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVy",
      "IdVersicherterAlt":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVyQWx0",
      "IkAlt": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElrQWx0"
    }
  ]
}
```

entsteht folgendes Ergebnis:

```
<Wert der Eigenschaft IdDatenlieferung> + | + <Wert des IdVersicherten Datensatz 1> + | + <Wert der IkAlt
Datensatz 1> + | + <Wert des IdDatensatzes Datensatz 2> + | + <Wert des IdVersicherterAlt Datensatz 2> + | +
<Wert der IkAlt Datensatz 2>

Hexadezimale Darstellung
32 30 32 34 2D 30 30 31 7C 32 30 32 34 2D 30 31 2D 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73
63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 7C
49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 69 63 68 65 72 74 65 72 20 57 65 72 74 20 64 65 72 20 49 6B 41
6C 74 7C 32 30 32 34 2D 30 31 2D 30 30 32 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C C3 BC 73
73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 7A 77 65 69 74 65 6E 20 49 64 56 65 72 73 69 63 68 65 72 74 65
72 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64
65 73 20 7A 77 65 69 74 65 6E 20 49 64 56 65 72 73 69 63 68 65 72 74 65 72 41 6C 74 7C 49 63 68 20 62 69 6E 20
65 69 6E 20 76 65 72 73 63 68 6C C3 BC 73 73 65 6C 74 65 72 20 57 65 72 74 20 64 65 73 20 7A 77 65 69 74 65 6E
20 49 6B 41 6C 74
```

Weitere Informationen zu der Bildung der Signatur finden sich unter dem Punkt Signierung von Eingangsdaten.

API für den Abruf von Fehlern, die bei der asynchronen Verarbeitung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten in der Vertrauensstelle aufgetreten sind

GET /notify/api/v1/insuranceupdatenotification/processingerrors

Diese Schnittstelle ist für den Abruf von Fehlern, die bei der asynchronen Verarbeitung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten in der Vertrauensstelle aufgetreten sind

Eingabedaten:

keine

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der StatusCode 204 zurückgegegben.

Wenn Daten zum Abruf vorhanden sind, wird der StatusCode 200 zurückgegeben. Als Content der Antwort wird die Liste der Datensätze zurückgegeben, bei denen es bei der asynchronen Verarbeitung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten auf Seiten der Vertrauensstellezu Fehlern kam. Elemente innerhalb dieser Liste sind vom Typ "InsuranceUpdateProcessingErrorsBundle".

Eigenschaft "Fehler"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "InsuranceUpdateProcessingErrorResult"
- Felder innerhalb eines Datensatzes:
  - IdDatenlieferung
    - definiert die für den meldenden Versicherungsträger eindeutige Id der Datenlieferung des Datensatzes
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen
  - IdDatensatz
    - definiert die eindeutige Id des Datensatzes (innerhalb der Datenlieferung)
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen
  - Code
    - definiert den Fehler, der aufgetreten ist, und wodurch der Datensatz nicht verarbeitet werden konnte
    - Optional: Nein
    - Typ: String
    - Schutz: Der Wert wird in Plaintext übertragen

Eigenschaft "Signatur"

- definiert die Signatur über die Daten dieser Datenlieferung
- Optional: Nein
- Format: Byte
- Typ: String

Fehlerfälle

Synchrone Fehlermeldungen

Fehler, die beim Abruf der Daten auftreten können, werden der meldenden Einrichtung in Form von HTTP-Statuscodes zurück gemeldet.
Liste der Fehlerfälle:

- StatusCode 401: dieser Fehler wird zurückgemeldet, wenn die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- StatusCode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- StatusCode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response

```
{
        "Fehler":
        [
            {
                "IdDatenlieferung": "2024-01",
                        "IdDatensatz": "2024-01-001",
                        "Code": "DecryptionError"
            },
            {
            "IdDatenlieferung": "2024-02",
                        "IdDatensatz": "2024-02-002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```

Signierung der Ausgangsdaten

Die Antwort, die durch die Aktion zurückgegeben werden, beinhalten neben der Liste der Daten auch die Signatur. Die Signatur wird dabei über die konkatenierten Werte innerhalb dieser Antwort gebildet.

Für beispielsweide einem JSON-Fragment mit diesen Daten

```
{
        "Fehler":
        [
            {
                "IdDatenlieferung": "2024-01",
                        "IdDatensatz": "2024-01-001",
                        "Code": "DecryptionError"
            },
            {
            "IdDatenlieferung": "2024-02",
                        "IdDatensatz": "2024-02-002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ]
}
```

entsteht folgendes Ergebnis:

```
<Wert IdDatenliegferung1> + | + <Wert IdDatensatz1> + | + <Wert Fehlercode1> + <Wert IdDatenliegferung2> + | +
<Wert IdDatensatz2> + | + <Wert Fehlercode2>

Hexadezimale Darstellung
32 30 32 34 2D 30 31 7C 32 30 32 34 2D 30 31 2D 30 30 31 7C 44 65 63 72 79 70 74 69 6F 6E 45 72 72 6F 72 7C 32
30 32 34 2D 30 32 7C 32 30 32 34 2D 30 32 2D 30 30 32 7C 57 72 6F 6E 67 46 6F 72 6D 61 74 49 64 56 65 72 73 69
63 68 65 72 74 65 72
```

Weitere Informationen finden sich unter dem Punkt Signierung der Ausgangsdaten.

Übergreifende Beispiele

In diesem Abschnitt wird anhand von Beispielen und Beispielaufrufen gezeigt, wie Daten, die an die Vertrauensstelle gesendet werden, verschlüsselt und signiert werden müssen. Auch wird hier beschrieben, wie die Registrierung an dem Dienst der Vertrauensstelle durchgeführt werden muss, sowie die Authentisierung und Authentifizierung an dem Dienst. Weiter wird hier auch gezeigt, wie Daten, die verschlüsselt von der Vertrauensstelle an ein Versicherungsträger gesendet werden, entschlüsselt werden können, sowie die Signatur eines Datenpaketes geprüft werden kann.

Verschlüsselung der Eingangsdaten

Bei der Verschlüsselung der Eingangsdaten wird sich an dem Konzept der gematik des E-Rezeptes angelehnt. Eine Beispielimplementierung zur Erstellung der verschlüsselten Datenfelder findet sich unter Adresse https://github.com/gematik/api-erp/blob/master/samples/VAUSimpleExample/VAU.cs (C#) bzw. unter https://github.com/gematik/api-erp/blob/master/samples/snippets/VAUClientCrypto.java (Java).

Bei der Verschlüsselung wird dabei für jedes zu verschlüsselnde Datenfeld ein eigenes temp. ECC-Schlüsselpaar gebildet. Dieses Schlüsselpaar dient dann als Eingangsparameter des ECDHBasicAgreement, mit dessen Hilfe gegen des öffentlichen Zertifikats des Empfängers (also die Vertrauensstelle oder die Registerstelle) das temp. SharedSecret gebildet wird. Dieses temp. SharedSecret wird als Eingangsparameter für die Schlüsselableitung (HKDF) verwendet. Als Info für die HKDF wird der fixe Wert "VST-IRD-Transport" verwendet. Der Wert, der durch die Schlüsselableitung gebildet wird, dient als Schlüssel für die Verschlüsselung des Eingangswertes mit dem Cipher AES-GCM.

Das Schlüsselmaterial der Vertrauensstelle lässt sich über einen gesonderten Downloadpunkt abrufen. Siehe hierzu Downloadpunkt für den Abruf von Schlüsselmaterial der Vertrauensstelle.

Nach Durchführung der Verschlüsselungsoperation wird der durch den Cipher gebildete Wert zusammen mit den öffentlichen Kurvenpunkten des temp. Schlüsselpaares als Wert innerhalb des JSON-Datenobjektes verwendet.

Ein Beispiel eines Datenkonstruktes nach Durchführung der Verschlüsselungsoperation und Bildung des zu übertragenen Wertes sieht wie folgt aus (hexadezimale Darstellung):

```
01 1E F6 00 88 48 EF 2D 4E AE 34 C3 5A 8A 16 8D 76 59 C0 84 F0 E1 EA DF 17 A5 33 3C B5 CF 4B B6 69 6A 5A C0 95
6C 59 A7 2E 95 93 F9 95 7C 46 E1 FC D2 9F A6 FF 19 AC 3E 30 69 0D 13 B5 57 48 87 87 2C DE 00 7A 78 83 74 D8 E2
ED 09 86 21 AC 15 C1 6B CE EB 27 44 A1 44 61 98 AA E5 3E 44 C1 30 6A 24 DE 46 46 BC 55

Das erste Byte bezeichnet dabei die Version. Diese ist nach der derzeitigen Implemantierung "1".
Die nächsten 32 Bytes definieren die X-Koordinate des temp. gebildeten Schlüsselpaares.
Die nächsten 32 Bytes definieren die Y-Koordinate des temp. gebildeten Schlüsselpaares.
Die nächsten 12 Bytes definieren den IV, der bei der Verschlüsselung AES-GCM verwendet wurde.
Die folgenden, letzten Bytes in dem Konstrukt definieren den AES-GCM-verschlüsselten Wert inklusive des
Authentication-Tags.
```

⚠️ für jeden verschlüsselten Wert muss ein eigenes, temporäres Schlüsselpaar gebildet und verwendet werden. Nach der Verschlüsselung eines Wertes kann dieses Schlüsselpaar wieder verworfen werden
⚠️ für jede AES-GCM-Verschüsselungsoperation muss ein eigener, zufällig gewürfelter IV gebildet werden

Signierung der Eingangsdaten

Die Signierung von Daten mit dem Schlüsselmaterial der TI ist nur mithilfe einer Aktion über den Konnektor bzw. des Basis-/KTR-Consumer möglich. Dazu wird ein entsprechender Endpunkt des Konnektors bzw. des Basis-/KTR-Consumer angesprochen, bei dem dann die aufgerufene SOAP-Acktion die eigentliche Arbeit übernimmt.

Request

URI	**https://192.168.x.y/Konnektorservice**
Method	POST
HTTP Header	`Content-Type: text/xml; charset=UTF-8` `SOAPAction: "http://ws.gematik.de/conn/SignatureService/v7.4#SignDocument"`

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns4="http://ws.gematik.de/conn/SignatureService/v7.4" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"> <soap:Body> <ns4:SignDocument> <ns2:CardHandle>CARDHANDLE</ns2:CardHandle> <ns3:Context> <ns2:MandantId>MANDANT_ID</ns2:MandantId> <ns2:ClientSystemId>CLIENTSYSTEM_ID</ns2:ClientSystemId> <ns2:WorkplaceId>WORKPLACE_ID</ns2:WorkplaceId> <ns2:UserId>USER_ID</ns2:UserId> </ns3:Context> <ns4:TvMode>NONE</ns4:TvMode> <ns4:JobNumber>SIG-001</ns4:JobNumber> <ns4:SignRequest RequestID='RequestID'> <ns4:OptionalInputs> <dss:SignatureType>urn:ietf:rfc:5652</dss:SignatureType> <ns4:IncludeEContent>false</ns4:IncludeEContent> </ns4:OptionalInputs> <ns4:Document ShortText='Dies ist eine zu signierende Nachricht' ID='RequestID'> <dss:Base64Data>VGVzdAo=</dss:Base64Data> </ns4:Document> <ns4:IncludeRevocationInfo>true</ns4:IncludeRevocationInfo> </ns4:SignRequest> </ns4:SignDocument> </soap:Body> </soap:Envelope>

In <dss:Base64Data></dss:Base64Data> befinden sich die zu signierenden Daten in Base64-Kodierung.
Die Signatur ist eine detatched nonQES - Signatur (nicht qualifizierte elektronische Signatur), die mithilfe des Signatur-Zertifikats auf der SMC-B-Karte gebildet wird. Da der Content separat in dem Datenpaket gesendet wird, wird dieser nicht in die Signatur eingebettet (<ns4:IncludeEContent>false</ns4:IncludeEContent>).
Der Signaturtyp muss CMS sein (urn:ietf:rfc:5652). Es wird eine ECDSA-Signatur benötigt.

Für den Zugriff auf die SMC-B für die Signaturerstellung muss die SMC-B durch die PIN freigeschaltet werden. Siehe hierzu Freigabe der SMC-B durch Eingabe der PIN.

Generelle Informationen zu den Schnittstellen des Konnektors bzw. Basis-/KTR-Consumer

Die korrekten Adressen der Endpunkte der Services des Konnektors bzw. Basis/KTR-Consumer sowie die zu verwendenden Version der Services lassen sich mithilfe der Herstellerinformationen bestimmen.

Die gematik-Spezifikation mit erweiterten Informationen lassen sich unter den Dokumenten "gemSpec_Basis_KTR_Consumer" bzw. "gemSpec_Kon" und der "gemSpec_Krypt" unter der Seite https://fachportal.gematik.de/dokumentensuche abrufen.

Generelle Informationen zu Parametern in SOAP-Requests

- Die Mandantenkonfiguration in den Parametern <ns3:Context></ns3:Context> zu Mandant, ClientSystem, Workplace und User muss entsprechend der festgelegten Werte des jeweiligen Versicherungsträgers angepasst werden.
- Das Handle der zu verwendenden Karte in <ns2:CardHandle></ns2:CardHandle> refernziert die zu verwendene Karte im Kartenterminal. Siehe hierzu Abfrage der Karten des Kartenterminals.

Prüfung eines TI-Zertifikates

Request

Method	POST

HTTP Header	Content-Type: text/xml; charset=UTF-8 SOAPAction: "http://ws.gematik.de/conn/CertificateService/v6.0#VerifyCertificate"
Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns4="http://ws.gematik.de/conn/CertificateServiceCommon/v2.0" xmlns:ns5="http://ws.gematik.de/conn/CertificateService/v6.0"> <soap:Body> <ns5:VerifyCertificate> <ns3:Context> <ns2:MandantId>_</ns2:MandantId> <ns2:ClientSystemId>_</ns2:ClientSystemId> <ns2:WorkplaceId>_</ns2:WorkplaceId> </ns3:Context> <ns4:X509Certificate>MIIFEjCCA /qgAwIBAgIHAXaCPYBSozANBgkqhkiG9w0BAQsFADCBmjELMAkGA1UEBhMCREUxHzAdBgNVBAoMFmdlbWF0aWsgR21iSCBOT1QtVkFMSUQxSD BGBgNVBAsMP0luc3RpdHV0aW9uIGRlcyBHZXN1bmRoZWl0c3dlc2Vucy1DQSBkZXIgVGVsZW1hdGlraW5mcmFzdHJ1a3R1cjEgMB4GA1UEAww XR0VNLlNNQ0ItQ0E0MSBURVNULU9OTFkwHhcNMjMwMzI4MDAwMDAwWhcNMjgwMzI3MjM1OTU5WjCB2TELMAkGA1UEBhMCREUxEDAOBgNVBAcM B0hhbWJ1cmcxDjAMBgNVBBEMBTIxMDM3MRowGAYDVQQJDBFTw7xkZXJxdWVyd2VnIDYzMzEsMCoGA1UECgwjUHJheGlzIERyLiBMaXNhIEfDt nJsaXR6ZXJOT1QtVkFMSUQxEzARBgNVBAQMCkfDtnJsaXR6ZXIxDTALBgNVBCoMBExpc2ExDDAKBgNVBAwMA0RyLjEsMCoGA1UEAwwjUHJheG lzIERyLiBMaXNhIEfDtnJsaXR6ZXJURVNULU9OTFkwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC3/t9dLrtLQblXh9 /OJ9UhA2VYup4WS+dPX+MTrccCpz1wP4DEBs6EzlXZtczlARlC+v4LOOP9KDEE91Hw2PLDi6aENQeV6Adiv40p5VFu9EW9XpOtaLdaWh6sLmh 2bI9VQmoM++PfeWiaKKKyXm44siNgg/OqX1QZlXBhAMkkDpWsl+SzW09eDGNC2O0B8OyUyv71cTj5djTUc/WMnQboU92E2fbd /9JLQF1Pbp7P1eLYO6MVFJV5QQiwg6l /Y6t7eWgta5uWAJ7aSbdUfT5vofwsIpYJY+7SNDNQ2U9qRqv9h8lEbuB4VCaZuuaiUevKvlYk3tvccveDKmRfw333AgMBAAGjggEaMIIBFjAg BgNVHSAEGTAXMAoGCCqCFABMBIEjMAkGByqCFABMBEwwHQYDVR0OBBYEFAgsYddT1CGX+EBGTCmyuvd2EfMzMAwGA1UdEwEB /wQCMAAwDgYDVR0PAQH /BAQDAgQwMDgGCCsGAQUFBwEBBCwwKjAoBggrBgEFBQcwAYYcaHR0cDovL2VoY2EuZ2VtYXRpay5kZS9vY3NwLzAfBgNVHSMEGDAWgBS6+DpY EfaG6gSibcR667mz+ktvwjBaBgUrJAgDAwRRME8wTTBLMEkwRzAWDBRCZXRyaWVic3N0w6R0dGUgQXJ6dDAJBgcqghQATAQyEyIxLVNNQy1CL VRlc3RrYXJ0ZS0tODgzMTEwMDAwMTQ3MzkxMA0GCSqGSIb3DQEBCwUAA4IBAQCB28Ci6ZXgepGu7QNvZz2+LPnRbKKpkkCYz /JX2ZpETvXdEHZ5g+60ffA7w5FtBCX+HiSFqoRuzZfEaPKOvA9hjrqNqe6hMF0vKc7sT1rwcldTIfnlrRzaIwGh55VrXoJsskM3V6wSPN8i9M P+UpOu02IxcF/aQobRiHobizqeSXrPvznoj0fqwN1Q5BNQbIGOfFOiOFd1rmbJdOxl /VBYC2nYbUTfbGqjJGONys+dCyD1+99fprWLg7uVAe7ozAs0fvfPxYJnxVfkH/CVJESB /s2mNL6BQgBzw0AESG3AwEHrfyU9X1dvEFfggzLNidsAyCY86iAVkY5dhzmWUo5H</ns4:X509Certificate> </ns5:VerifyCertificate> </soap:Body> </soap:Envelope>

Response

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <ns4:VerifyCertificateResponse xmlns:ns9="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:ns8="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:ns7="http://www.w3.org/2000/09/xmldsig#" xmlns:ns6="http://ws.gematik.de/conn /CertificateServiceCommon/v2.0" xmlns:ns5="http://ws.gematik.de/tel/error/v2.0" xmlns:ns4="http://ws.gematik.de/conn/CertificateService /v6.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"> <ns2:Status> <ns2:Result>OK</ns2:Result> </ns2:Status> <ns4:VerificationStatus> <ns4:VerificationResult>VALID</ns4:VerificationResult> </ns4:VerificationStatus> <ns4:RoleList> <ns4:Role>1.2.276.0.76.4.50</ns4:Role> </ns4:RoleList> </ns4:VerifyCertificateResponse> </soap:Body> </soap:Envelope>

Freigabe der SMC-B durch Eingabe der PIN

Request

Method	POST
HTTP Header	`Content-Type: text/xml; charset=UTF-8` `SOAPAction: "http://ws.gematik.de/conn/CardService/v8.1#VerifyPin"`

Payload	`<soap:Envelope`
	`xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"`
	`xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"`
	`xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0"`
	`xmlns:ns4="http://ws.gematik.de/conn/CardServiceCommon/v2.0"`
	`xmlns:ns5="http://ws.gematik.de/conn/CardService/v8.1">`
	`<soap:Body>`
	`<ns5:VerifyPin>`
	`<ns3:Context>`
	`<ns2:MandantId>MANDANT_ID</ns2:MandantId>`
	`<ns2:ClientSystemId>CLIENTSYSTEM_ID</ns2:ClientSystemId>`
	`<ns2:WorkplaceId>WORKPLACE_ID</ns2:WorkplaceId>`
	`</ns3:Context>`
	`<ns2:CardHandle>CARDHANDLE</ns2:CardHandle>`
	`<ns4:PinTyp>PIN.SMC</ns4:PinTyp>`
	`</ns5:VerifyPin>`
	`</soap:Body>`
	`</soap:Envelope>`

Abfrage der Karten des Kartenterminals

Request

URI	**https://192.168.x.y/Konnektorservice**
Method	POST
HTTP Header	`Content-Type: text/xml; charset=UTF-8` `SOAPAction: "http://ws.gematik.de/conn/EventService/v7.2#GetCards"`

Payload	`<soap:Envelope`
	`xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"`
	`xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"`
	`xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0"`
	`xmlns:ns4="http://ws.gematik.de/conn/EventService/v7.2">`
	`<soap:Body>`
	`<ns4:GetCards>`
	`<ns3:Context>`
	`<ns2:MandantId>MANDANT_ID</ns2:MandantId>`
	`<ns2:ClientSystemId>CLIENTSYSTEM_ID</ns2:ClientSystemId>`
	`<ns2:WorkplaceId>WORKPLACE_ID</ns2:WorkplaceId>`
	`</ns3:Context>`
	`</ns4:GetCards>`
	`</soap:Body>`
	`</soap:Envelope>`

Response

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <ns4:GetCardsResponse xmlns:ns12="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:ns11="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:ns10="http://www.w3.org/2000/09/xmldsig#" xmlns:ns9="http://ws.gematik.de/conn/CardTerminalInfo/v8.0" xmlns:ns8="http://ws.gematik.de/int/version/ProductInformation/v1.1" xmlns:ns7="http://ws.gematik.de/conn/CardService/v8.1" xmlns:ns6="http://ws.gematik.de/conn/CardServiceCommon/v2.0" xmlns:ns5="http://ws.gematik.de/tel/error/v2.0" xmlns:ns4="http://ws.gematik.de/conn/EventService/v7.2" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"> <ns2:Status> <ns2:Result>OK</ns2:Result> </ns2:Status> <ns7:Cards> <ns7:Card> <ns2:CardHandle>CARDHANDLE</ns2:CardHandle> <ns6:CardType>SMC-B</ns6:CardType> <ns7:CardVersion> <ns7:COSVersion> <ns7:Major>4</ns7:Major> <ns7:Minor>6</ns7:Minor> <ns7:Revision>0</ns7:Revision> </ns7:COSVersion> <ns7:ObjectSystemVersion> <ns7:Major>4</ns7:Major> <ns7:Minor>8</ns7:Minor> <ns7:Revision>0</ns7:Revision> </ns7:ObjectSystemVersion> </ns7:CardVersion> <ns6:Iccsn>xxx</ns6:Iccsn> <ns6:CtId>xxx</ns6:CtId> <ns6:SlotId>xx</ns6:SlotId> <ns7:InsertTime>xxxx</ns7:InsertTime> <ns7:CardHolderName>Praxis Dr. Mustermann TEST-ONLY</ns7:CardHolderName> <ns7:CertificateExpirationDate>xxx</ns7:CertificateExpirationDate> </ns7:Card> <ns7:Card> ... </ns7:Card> </ns7:Cards> </ns4:GetCardsResponse> </soap:Body> </soap:Envelope>

Abfrage des AUT-Zertifikates der SMC-B-Karte

Request

URI	**https://192.168.x.y/Konnektorservice**
Method	POST
HTTP Header	Content-Type: text/xml; charset=UTF-8 SOAPAction: "http://ws.gematik.de/conn/CertificateService/v6.0#ReadCardCertificate"

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns4="http://ws.gematik.de/conn/CertificateService/v6.0"> <soap:Body> <ns4:ReadCardCertificate> <ns2:CardHandle>CARDHANDLE</ns2:CardHandle> <ns3:Context> <ns2:MandantId>MANDANT_ID</ns2:MandantId> <ns2:ClientSystemId>CLIENTSYSTEM_ID</ns2:ClientSystemId> <ns2:WorkplaceId>WORKPLACE_ID</ns2:WorkplaceId> </ns3:Context> <ns4:CertRefList> <ns4:CertRef>C.AUT</ns4:CertRef> </ns4:CertRefList> <ns4:Crypt>ECC</ns4:Crypt> </ns4:ReadCardCertificate> </soap:Body> </soap:Envelope>

Response

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <ns4:ReadCardCertificateResponse xmlns:ns9="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:ns8="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:ns7="http://www.w3.org/2000/09/xmldsig#" xmlns:ns6="http://ws.gematik.de/conn/CertificateServiceCommon/v2.0" xmlns:ns5="http://ws.gematik.de/tel/error/v2.0" xmlns:ns4="http://ws.gematik.de/conn/CertificateService/v6.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"> <ns2:Status> <ns2:Result>OK</ns2:Result> </ns2:Status> <ns6:X509DataInfoList> <ns6:X509DataInfo> <ns6:CertRef>C.AUT</ns6:CertRef> <ns6:X509Data> <ns6:X509IssuerSerial> <ns6:X509IssuerName>CN=GEM.SMCB-CA51 TEST-ONLY,OU=Institution des Gesundheitswesens-CA der Telematikinfrastruktur,O=gematik GmbH NOT-VALID,C=DE</ns6:X509IssuerName> <ns6:X509SerialNumber>Seriennummer des Zertifikates</ns6:X509SerialNumber> </ns6:X509IssuerSerial> <ns6:X509SubjectName>CN=xxx,O=xxx,L=xxx,C=DE</ns6:X509SubjectName> <ns6:X509Certificate>Zertifikat_Base64_codiert</ns6:X509Certificate> </ns6:X509Data> </ns6:X509DataInfo> </ns6:X509DataInfoList> </ns4:ReadCardCertificateResponse> </soap:Body> </soap:Envelope>

Operation ExternalAuthenticate

Request

URI	**https://192.168.x.y/Konnektorservice**
Method	POST
HTTP Header	Content-Type: text/xml; charset=UTF-8 SOAPAction: "http://ws.gematik.de/conn/SignatureService/v7.4#ExternalAuthenticate"

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns4="http://ws.gematik.de/conn/SignatureService/v7.4" xmlns:dss="urn:oasis:names:tc:dss:1.0:core:schema"> <soap:Body> <ns4:ExternalAuthenticate> <ns2:CardHandle>CARDHANDLE</ns2:CardHandle> <ns3:Context> <ns2:MandantId>MANDANT_ID</ns2:MandantId> <ns2:ClientSystemId>CLIENTSYSTEM_ID</ns2:ClientSystemId> <ns2:WorkplaceId>WORKPLACE_ID</ns2:WorkplaceId> </ns3:Context> <ns4:OptionalInputs> <dss:SignatureType>urn:bsi:tr:03111:ecdsa</dss:SignatureType> </ns4:OptionalInputs> <ns4:BinaryString> <dss:Base64Data MimeType="application/octet-stream">lO4FkzXlh+UBzEv5BhPggU8Ap7CLx8ZI /YZaKvaiLMI=</dss:Base64Data> </ns4:BinaryString> </ns4:ExternalAuthenticate> </soap:Body> </soap:Envelope>

Response

Payload	<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <ns5:ExternalAuthenticateResponse xmlns:ns15="urn:oasis:names:tc:SAML:1.0:assertion" xmlns:ns14="http://ws.gematik.de/conn/CertificateServiceCommon/v2.0" xmlns:ns13="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:ns12="http://www.w3.org/2001/04/xmlenc#" xmlns:ns11="urn:oasis:names:tc:dss-x:1.0:profiles:SignaturePolicy:schema#" xmlns:ns10="http://uri.etsi.org/02231/v2#" xmlns:ns9="urn:oasis:names:tc:dss-x:1.0:profiles:verificationreport:schema#" xmlns:ns8="http://uri.etsi.org/01903/v1.3.2#" xmlns:ns7="http://ws.gematik.de/tel/error/v2.0" xmlns:ns6="http://www.w3.org/2000/09/xmldsig#" xmlns:ns5="http://ws.gematik.de/conn/SignatureService/v7.4" xmlns:ns4="urn:oasis:names:tc:dss:1.0:core:schema" xmlns:ns3="http://ws.gematik.de/conn/ConnectorContext/v2.0" xmlns:ns2="http://ws.gematik.de/conn/ConnectorCommon/v5.0"> <ns2:Status> <ns2:Result>OK</ns2:Result> </ns2:Status> <ns4:SignatureObject> <ns4:Base64Signature Type="urn:bsi:tr:03111:ecdsa">MEQCIBZUl+5f0iFAoX558 /k+9HNPnCvDoKreSor+QSn4lsd6AiAhdgrjuMxNz9d1w1BjAIOsMSlZfSdJ/q/5PT5duYzG5g==</ns4:Base64Signature> </ns4:SignatureObject> </ns5:ExternalAuthenticateResponse> </soap:Body> </soap:Envelope>

Downloadpunkt für den Abruf von Schlüsselmaterial der Vertrauensstelle

Schlüsselmaterial, das von der Vertrauensstelle verwendet wird, lässt sich über den gesonderten Downloadpunkt XXXXXXXXX/.wellknown herunterladen. Bei den bereitgestellten Schlüsseln handelt es sich um Elliptic Curve-Schlüssel, von denen der öffentliche Schlüssel als X und Y innerhalb des signierten JWT-Token enthalten ist.

Das JWT-Token ist mit der ECDSA-Signatur versehen, die mithilfe des AUT-Zertifikats (TI-Zertifikat) der Vertrauensstelle gebildet wurde. Dieses Zertifikat befindet sich als x5c-Angabe innerhalb des Headers des JWT.
⚠️ vor der Verwendung des JWT-Token (bzw. der darin enthaltenen Schlüssel) muss die Gültigkeit des Tokens überprüft werden. Dazu zählen die zeitliche Gültigkeit des Tokens und die Korrektheit der Signatur. Außerdem muss das im Token angegebene Signaturzertifikat der TI geprüft werden (siehe hierzu Prüfung eines TI-Zertifikates).

Nichtfunktionale Festlegungen

aktuell in Klärung

Test

Definition von Testdaten

Die Regeln für die Testdaten stellen sicher, dass patienten-identifizierende Daten nicht in Testumgebungen (für IRD die Referenzumgebung der TI) verwendet werden. Hierbei sind die speziellen Regeln für die Referenzumgebung der TI und die Produktivumgebung der TI zu beachten[2] [3].

Testdaten Krankenversichertennummer (KVNR)

Der GKV-SV hat einen KVNR-Nummernkreis bereitgestellt, der ausschließlich für Tests im Rahmen des Implantateregisters Deutschland (IRD) genutzt werden darf. Die KVNRs dieses Nummernkreises werden gesichert keiner Person zugeordnet.

Definition: Nummernkreis A1111xxxxP , wobei x für eine Ziffer von 0 bis 9 und P für die Prüfziffer steht.

Fußnoten

1.) Die Angabe der speziellen Krankenversichertennummer (KVNR) bei Meldungen für Patienten ohne deutsche Krankenversicherung werden nicht akzeptiert.

2.) Für die Verwendung von KVNR gelten folgende Einschränkungen:
Referenzumgebung der TI: In dieser Umgebung dürfen nur vordefinierte Nummern aus dem Testnummernkreis (Definition Testdaten) verwendet werden. Bei der Verwendung von prod. KVNR-Nummern in der Referenz-Umgebung wird die Meldung nicht angenommen und dem Sender ein Fehler zuückgegeben.

Produktivumgebung der TI: In dieser Umgebung werden nur produktive Daten erwartet. Eine Verwendung einer produktiven KVNR, d.h. einer potentiell einem Versicherten zugeordneten KVNR, als Testdatensatz ist in dieser Umgebung verboten. Als Ausnahme sind Tests der korrekten Anbindung der meldenden Einrichtung an die Produktivumgebung der TI ausschließlich mit einer KVNR aus dem Testnummernkreis (Definition Testdaten) zugelassen.

Kontakt

Robert Koch-Institut
Nordufer 20
13353 Berlin

E-Mail: Vertrauensstelle-IRD@rki.de