IRD Technische Spezifikation für Krankenversicherungsträger (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	• bei den Meldungen Vitalstatus und Versicherungswechsel wird nur der grundsätzliche HTTP-Statuscode zurückgegeben • die Liste zu fehlerhaften Datensätzen entfällt; diese Fehler werden jetzt asynchron zur Abholung bereitgestellt	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.3	28.06.2024	verbindliche Version für die API Version 1.0 • Anmeldetoken spezifiziert • einmalige Registrierung bei der Vertrauensstelle angepasst • Downloadpunkt für den public key der Vertrauensstelle beschrieben • Prüfung der Gültigkeit der Schlüssel und Zertifikate der Vertrauensstelle beschrieben • Informationen zum patienten-identifizierenden Datum ergänzt • Sitzungsschlüssel spezifiziert	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.4	19.07.2024	aktualisierte verbindliche Version für die API Version 1.0 • Formatierungen und Links angepasst • Hinweis zu JSON Parser Konformität • Detailiertes Beispiel zum Anmelde-Token • Beispiel zu verschlüsselten Ausgangsdaten • Format Id-Nummer Heilfürsorge BW	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.5	09.09.2024	aktualisierte verbindliche Version für die API Version 1.0 • Authentisierung und Authentifizierung mit custom HTTP Authorization Header: Reduktion auf das bei der Registrierung vorab angegebene Institutionskennzeichen (IK).	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.6	30.09.2024	▪ Anpassung der Festlegung Versichertenwechsel in HTTP Schnittstelle ▪ IkMelder (ehemals IkAlt) als redundantes Feld entfernt, da es dem IK in der Anmeldeinformation entsprechen muss und damit festgelegt ist, dass der authentisierte KVT einen Kassenwechsel meldet ▪ Festlegung Verschlüsselung eingehende und ausgehende Daten: ECIES in Anlehnung anderer Fachverfahren in der TI. ▪ Umstrukurierung und redaktionelle Überarbeitung	RKI Referat VIG (Vertrauensstelle-ird@rki.de)
1.7	09.10.2024	Korrektur HTTP-Methoden „GET“ zu „POST“: • POST /notify/api/v1/vitalstatusnotification/requests • POST /notify/api/v1/anonymizationnotifications • POST /notify/api/v1/vitalstatusnotification/processingresults/ und • POST /notify/api/v1/insuranceupdatenotification/processingresults/	RKI Referat VIG (Vertrauensstelle-ird@rki.de)

1.8	01.11.2024	• Ergänzung von Informationen zur Signierung der Eingangsdaten (KVT-> VST)
		  ○ Signatur Anmelde-Token: <ns4:IncludeEContent>true</ns4:IncludeEContent>
		  ○ Signatur Payload: <ns4:IncludeEContent>false</ns4:IncludeEContent>
		• Ergänzung von Informationen zum SessionKey für die Verschlüsselung der Rückgabedaten (VST-> KVT)
		• Rückgabe des Institutionskennzeichens (IK) als HTTP-Response Header Eintrag in allen Replies
		• Korrektur Beschreibung Typ der Antwort bei
		  ○ POST /notify/api/v1/anonymizationnotifications
		  ○ POST /notify/api/v1/vitalstatusnotification/processingresults
		  ○ POST /notify/api/v1/insuranceupdatenotification/processingresults
		• Ergänzung leerer Content der Antwort, wenn bei einer Datenlieferung alle Datensätze in Ordnung sind, bei
		  ○ POST /notify/api/v1/vitalstatusnotification/processingresults
		  ○ POST /notify/api/v1/insuranceupdatenotification/processingresults
			RKI Referat VIG (Vertrauensstelle-ird@rki.de)

Inhalt

Einleitung

Die Vertrauensstelle des Implantateregisters Deutschland (IRD) beim Robert Koch-Institut (RKI) stellt Schnittstellen zur Meldung von Vitalstatus und Versicherungswechseln für die Krankenversicherungsträger (KVT) bereit.

Diese technische Spezifikation beschreibt die Kommunikation eines meldenden Krankenversicherungsträgers für den Datenaustausch über REST-Schnittstellen mit 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 Krankenversicherungsträgers dieses Personenkreises. Als Krankenversicherungsträger 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.

Von der Vertrauensstelle IRD werden für den Produktivbetrieb jeweils die aktuell veröffentlichte Version der Schnittstelle sowie die vorhergehende Version unterstützt.

Allgemeine Informationen

Die API Version 1.0 akzeptiert als patienten-identifizierendes Datum den unveränderbaren 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. Für alle meldepflichtigen KVT werden zwei mögliche Nummernkreise einer gültigen Identifikationsnummer unterstützt:

1. Der unveränderbare Teil der Krankenversichertennummer (KVNR) für alle gesetzlichen und privaten Krankenversicherungsträger, einschließlich der Heilfürsorge Bundespolizei gemäß der Richtlinie nach § 290 SGB V (siehe https://gkv-datenaustausch.de/kvnr/kvnr.jsp).
2. Die unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer im Format der Steueridentifikationsnummer entsprechend Identifikationsnummerngesetz (IDNrG) für die Heilfürsorge Bundeswehr. Eine Beschreibung des Aufbaus und Algorithmus zur Berechnung der Prüfziffer findet sich im Dokument "Prüfung der Steuer- und Steueridentifikationsnummer sowie der Ordnungskriterien bei der Grundsteuer" unter https://download.elster.de/download/schnittstellen/Pruefung_der_Steuer_und_Steueridentifikatsnummer.pdf (Kapitel 2.2 Aufbau der Steueridentifikationsnummer (IdNr) Seite 6 ff.; Version mit Stand 3. September 2024).

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 bei der Vertrauensstelle

Ein Krankenversicherungsträger muss sich vor dem ersten Aufruf einer API-Funktion einmalig bei der Vertrauensstelle mit dem Institutionskennzeichen (IK) nach § 293 SGB V und der Telematik-Id (TID) registrieren. Bei relevanten Änderungen dieser Registrierungsdaten ist eine erneute Registrierung bei der Vertrauensstelle erforderlich.

Für die Registrierung stellt die Vertrauensstelle unterhalb der URL https://xml.ir-d.de/rst/download/vst/krankenversicherungstraeger/ das Formular *Kranken versicherungstraeger_Registrierung_IRD.pdf* zur Verfügung, das ausgefüllt an die Vertrauensstelle zu senden ist.

Angabe bei mehreren IKs

Krankenversicherungsträger mit mehreren Institutionskennzeichen verwenden zur Registrierung das sog. Haupt-IK (s. Spezifikation unter https://www.gkv-datenaustausch.de/faq/faq.jsp "Rundschreiben zum Institutionskennzeichen").

Info zur Telematik-ID

Zur Info: Der Aufbau der Telematik-ID ist hier spezifiziert: https://gemspec.gematik.de/docs/gemSpec/gemSpec_PKI/latest/#4.7.2 . Die Telematik-ID wird im Feld registrationNumber der Extension Admission im X.509-Zertifikat (hier konkret des C.HCI.AUT) hinterlegt.

Authentisierung und Authentifizierung

Bei jedem Aufruf einer API-Funktion ist ein Anmelde-Token als HTTP Header Authorization vom Typ "custom" zu senden. Das bedeutet, das das Anmeldetoken für jeden Aufruf immer neu zu bilden ist.

Dieses AnmeldeToken wird folgendermaßen gebildet:

- Als Format wird CMS gemäß RFC5652 verwendet, als Ergebnis der Signatur mit dem C.HCI.AUT x.509-Zertifikat des übermittelnden Krankenversicherungsträgers (am Basis-Consumer/Konnektor).
- Es muss eine ECDSA Signatur erzeugt werden (RSA wird ab 2025 nicht mehr unterstützt).
- Der zu signierende Wert ist das bei der Registrierung angegebene Instititutionskennzeichen (IK) des KVT. Dieses IK muss mit dem bei der Registrierung des KVT übereinstimmen.
- Der Inhalt – also das registrierte IK – muss in das CMS eingebettet werden.
- Bei der Prüfung durch die Vertrauensstelle ist eine maximale Abweichung des Zeitstempels der Signatur vom Absender (KVT) bei Empfang in der VST von 60 Sekunden zulässig.
  Dieser Zeitstempel findet sich in dem CMS Container unter object: signingTime (1.2.840.113549.1.9.5). Das Format ist UTC / GMT Zeitzone.
  Hinweis: Die Clients müssen die Zeitsynchronisation gemäß den Vorgaben der Telematikinfrastruktur nutzen. Siehe dazu: https://gemspec.gematik.de/docs/gemSpec/gemSpec_Net/latest/#6.

Das Ergebnis ist dann ein Base64 kodierter PKCS#7 Container (CMS). Die Telematik-ID wird durch die VST aus dem im CMS enthaltenen x.509 Zertifikat C.HCI.AUT entnommen (s. https://gemspec.gematik.de/docs/gemSpec/gemSpec_PKI/latest/#4.7). Im CMS befindet sich diese Information unter object: Professional Information or basis for Admission (1.3.36.8.3.3).

IK und Telematik-ID müssen mit den bei der Registrierung mitgeteilten Daten übereinstimmen. Sollten sich diese Identifikatoren ändern, muss dies rechtzeitig durch einen Änderungsantrag der Vertrauensstelle bekannt gemacht werden.

Ein Beispiel dazu:

1. Erstellen der Eingangsdaten für die Signatur am Konnektor mit dem C.HCI.AUT Zertifikat des KVT.

```
####
# Header data - das Haupt-IK des meldenden KVT:
# IK        := 104127692
##

# Es wird die ASCII Repräsentation verwendet
00000000  31 30 34 31 32 37 36 39  32   |104127692|

# Der Aufruf für die Signatur am Konnektor erwartet einen Base64 kodierten Eingabewert.
# 104127692 base64 kodiert hat diesen Inhalt:
00000000  4d 54 41 30 4d 54 49 33  4e 6a 6b 79              |MTA0MTI3Njky|


Der Eingabewert für die SOAPAction SignDocument des Basis Consumer lautet also:
MTA0MTI3Njky
```

2. Erstellen der ECDSA Signatur am Konnektor. Ein Beispiel findet sich unter Signatur Dienst des Konnektors.
3. Ergebnis der erstellten ECDSA Signatur (eine p7s Datei nach RFC 2311 – application/pkcs7-signature .p7s)

Beispiel CMS (PKCS7 Container)

```
MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgEFADCABgkqhkiG9w0B
BwGggCSABAkxMDQxMjc2OTIAAAAAAACggDCCA4YwggMtoAMCAQICBwF/9GLesocw
CgYIKoZIzj0EAwIwgZoxCzAJBgNVBAYTAkRFMR8wHQYDVQQKDBZnZW1hdGlrIEdt
YkggTk9ULVZBTElEMUgwRgYDVQQLDD9JbnN0aXR1dGlvbiBkZXMgR2VzdW5kaGVp
dHN3ZXNlbnMtQ0EgZGVyIFRlbGVtYXRpa2luZnJhc3RydWt0dXIxIDAeBgNVBAMM
F0dFTS5TTUNCLUNBNTEgVEVTVC1PTkxZMB4XDTIzMDMyODAwMDAwMFoXDTI4MDMy
NzIzNTk1OVowgdkxCzAJBgNVBAYTAkRFMRAwDgYDVQQHDAdIYW1idXJnMQ4wDAYD
VQQRDAUyMTAzNzEaMBgGA1UECQwRU8O8ZGVycXVlcndlZyA2MzMxLDAqBgNVBAoM
I1ByYXhpcyBEci4gTGlzYSBHw7ZybGl0emVyTk9ULVZBTElEMRMwEQYDVQQEDApH
w7ZybGl0emVyMQ0wCwYDVQQqDARMaXNhMQwwCgYDVQQMDANEci4xLDAqBgNVBAMM
I1ByYXhpcyBEci4gTGlzYSBHw7ZybGl0emVyVEVTVC1PTkxZMFowFAYHKoZIzj0C
AQYJKyQDAwIIAQEHA0IABKH5mkRn/xH5zdQE+xnQ6S6tdQF4xp7PEADuVQo83pe5
dhr0+d9L6AttvmCt5Y+fbVPilzRjywIRNJNuZPh1YrCjggEaMIIBFjAdBgNVHQ4E
FgQUCl2pGZ6ALqe9xfQHnS0ZkjUXLmAwDAYDVR0TAQH/BAIwADA4BggrBgEFBQcB
AQQsMCowKAYIKwYBBQUHMAGGHGh0dHA6Ly9laGNhLmdlbWF0aWsuZGUvb2NzcC8w
DgYDVR0PAQH/BAQDAgZAMB8GA1UdIwQYMBaAFAaY6QJV/8mfXKNlDvFd4iD1hPuT
MCAGA1UdIAQZMBcwCgYIKoIUAEwEgSMwCQYHKoIUAEwETjBaBgUrJAgDAwRRME8w
TTBLMEkwRzAWDBRCZXRyaWVic3N0w6R0dGUgQXJ6dDAJBgcqghQATAQyEyIxLVNN
Qy1CLVRlc3RrYXJ0ZS0tODgzMTEwMDAwMTQ3MzkxMAoGCCqGSM49BAMCA0cAMEQC
IAwOHjakFJYefftNL7jyDdweYv0lIbQ5TZKNB831u0gEAiBckW8iKwUCyJv2tA4t
k/vhvEir6B/XmfIIC/VLvojY9AAAMYIDAjCCAv4CAQEwgaYwgZoxCzAJBgNVBAYT
AkRFMR8wHQYDVQQKDBZnZW1hdGlrIEdtYkggTk9ULVZBTElEMUgwRgYDVQQLDD9J
bnN0aXR1dGlvbiBkZXMgR2VzdW5kaGVpdHN3ZXNlbnMtQ0EgZGVyIFRlbGVtYXRp
a2luZnJhc3RydWt0dXIxIDAeBgNVBAMMF0dFTS5TTUNCLUNBNTEgVEVTVC1PTkxZ
AgcBf/Ri3rKHMA0GCWCGSAFlAwQCAQUAoIIB6TAYBgkqhkiG9w0BCQMxCwYJKoZI
hvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0yNDA5MjYxNjEwNTVaMCQGBgQAjUUCATEa
DBhhcHBsaWNhdGlvbi9vY3RldC1zdHJlYW0wKgYJKoZIhvcNAQk0MR0wGzANBglg
hkgBZQMEAgEFAKEKBggqhkjOPQQDAjAvBgkqhkiG9w0BCQQxIgQgrCkPCAGHxqPC
U189mC1dJqaSBTrx/2d9Wr/WfLe1FcowMAYLKoZIhvcNAQkQAgQxITAfDBJDdXN0
b21BdXRoVG9rZW5LVlQGCSqGSIb3DQEHATCB+QYLKoZIhvcNAQkQAi8xgekwgeYw
geMwgeAwDQYJYIZIAWUDBAIBBQAEIHLLGO+RRsV5aaRH3hJoBXwFpnAw5yW5mcdw
WJdtIJciMIGsMIGgpIGdMIGaMQswCQYDVQQGEwJERTEfMB0GA1UECgwWZ2VtYXRp
ayBHbWJIIE5PVC1WQUxJRDFIMEYGA1UECww/SW5zdGl0dXRpb24gZGVzIEdlc3Vu
ZGhlaXRzd2VzZW5zLUNBIGRlciBUZWxlbWF0aWtpbmZyYXN0cnVrdHVyMSAwHgYD
VQQDDBdHRU0uU01DQi1DQTUxIFRFU1QtT05MWQIHAX/0Yt6yhzAKBggqhkjOPQQD
AgRGMEQCIBKQ+B++WRViHJPMidps6HHyqZNcnx0AkB4EHBz0Ki1kAiBWj5P7f5hX
VvoB4n3+ZQ2r1Rg4NdcPv0NEZ077sn2vo6EAAAAAAAAA
```

4. Übermitteln des CMS im HTTP Header Authorization

Signierter Wert im HTTP-Header Authorization übermitteln

```
# Beispiel des HTTP Headers mit (gekürztem) CMS von oben, das vom Basis Consumer mit Aufruf SignDocument
als Ergebnis erzeugt wurde:

Host: vst-ird-ru.rki-ti.de

Content-Length
Authorization: Custom MIAGCSqGSIb3D...sn2vo6EAAAAAAAAA
...
```

Downloadpunkt für den Abruf von Schlüsselmaterial der Vertrauensstelle

Die Vertrauensstelle Implantateregister Deutschland besitzt eine digitale Identität in der Telematikinfrastruktur und ist Teilnehmer in der Komponenten-PKI der TI. Als weitere Anwendung für den Datenaustausch in der Telematikinfrastruktur (WANDA) ist das Schlüsselmaterial jedoch nicht allgemein im Verzeichnisdienst (VZD) auffindbar bzw. abrufbar. Daher werden öffentliche Schlüssel des Fachdienstes Vertrauensstelle Implantateregister als x.509 Zertifikate an einem Downloadpunkt bereitgestellt. Siehe hierzu Kapitel *5.9 FD – Fachanwendungsspezifische Dienste* im Dokument *Übergreifende Spezifikation PKI* der gematik. Für die zum Zeitpunkt dieser Spezifikation aktuelle Version 2.18.0 ist unter https://gemspec.gematik.de/docs/gemSpec/gemSpec_PKI/latest/#5.9 abrufbar.

Das im Kontext Meldung Vitalstatus und Kassenwechsel zu verwendende Schlüsselmaterial der Vertrauensstelle steht daher für die PU unter https://vst-ird.rki-ti.de/pub_keys zum Download bereit. Für die RU sind die Schlüssel unter https://vst-ird-ru.rki-ti.de/TEST-ONLY/pub_keys abrufbar. Die X.509-Zertifikate sind DER-kodiert. Für den Abruf der Zertifikate muss der Content-Type: application/pkix-cert [RFC-2582] angegeben werden.

Folgende x509 Zertifikate aus der Komponenten PKI der Telematikinfrastruktur stehen zum Download zur Verfügung:

- ENC.der (dies entspricht dem Zertifikatstyp C.FD.ENC)
- AUT.der (dies entspricht dem Zertifikatstyp C.FD.AUT)
- SIG.der (dies entspricht dem Zertifikatstyp C.FD.SIG)

Diese entsprechen den sog. X.509 Zertifikatsprofilen der Fachanwendungsspezifischen Dienste (s. https://gemspec.gematik.de/docs/gemSpec/gemSpec_PKI/latest/#5.9.3)

Beispiel mit curl für das AUT-Zertifikat.

RU

> curl --header "Content-Type: application/pkix-cert" --output vst-ird-ru-aut.crt https://vst-ird-ru.rki-ti.de/pub_keys/TEST-ONLY/AUT.der

PU

> curl --header "Content-Type: application/pkix-cert" --output vst-ird-aut.crt https://vst-ird.rki-ti.de/pub_keys/AUT.der

Temporär lokal gespeicherte Schlüssel der Vertrauensstelle sind mindestens nach 24 Stunden erneut vom Downloadpunkt abzurufen und die Gültigkeit der Zertifikate zu prüfen (siehe hierzu Prüfung eines TI-Zertifikates).

Information zu öffentlichen Schlüsseln der Registerstelle

Schlüsselmaterial der Registerstelle liegt nicht in Verantwortung der Vertrauensstelle. Den Abruf benötigter Schlüssel müssen die KVT also mit der Registerstelle abstimmen.

Datenformate

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 unbekannter Eigenschaften führt zu einem Fehler.

Hinweis: RFC 8259 (https://www.rfc-editor.org/rfc/rfc8259) definiert Arrays als *ordered sequence of zero or more values*. In der Implementierung verwendete Parser sind auf entsprechende Konformität zu prüfen.

Rückgabe

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

Signierung von Daten

Signierung der Eingangsdaten (KVT-> VST)

Signaturen der Daten, die von den Krankenversicherungsträgern mit den jeweiligen Daten mitzusenden sind, gewährleisten die Integrität und die Authentizität der übertragenen Daten. Es wird hier das Schema Encrypt-Then-Sign angewendet, die Signatur wird also über die bereits verschlüsselten und danach Base64 kodierten Daten gebildet.

Bei der Signatur handelt es sich um eine nonQES - Signatur (nicht qualifizierte elektronische Signatur). Hier wird bei der Signaturbildung der Content nicht mit eingebettet!

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

Für beispielsweise ein JSON-Fragment mit diesen Daten

```
{
  "IdDatenlieferung": "2024-01",
"Meldungen": [
{
      "IdDatensatz": "8-0000001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlcyBUb2Rlc2RhdHVt"
},
{
      "IdDatensatz": "8-0000002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVzIFRvZGVzZGF0dW0"
}
]
}
```

entsteht folgendes Ergebnis:

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

Hexadezimale Darstellung
32 30 32 34 2D 30 31 7C 38 2D 30 30 30 30 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68
6C FC 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 FC 73 73 65 6C 74 65 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 76 65 72 73 63 68 6C FC 73 73 65 6C 74 65 73 20 54 6F 64 65 73 64 61 74 75
6D 7C 38 2D 30 30 30 30 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 FC 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 FC 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 FC 73 73 65 6C 74 65 73 20 54 6F 64 65 73 64 61 74 75 6D
```

Die Plain Bytes werden Base64 kodiert an die Schnittstelle mit SOAPAction SignDocument 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 APIs.

Signierung des SessionKeys (KVT -> VST)

In den Abruffunktionen *Benachrichtigung über Anonymisierung* und *Anfragen zu Vitalstatusmeldungen* sendet der Krankenversicherungsträger einen SessionKey

an die Vertrauensstelle. Mit diesem SessionKey verschlüsselt die Vertrauensstelle die Rückgabewerte für den Krankenversicherungsträger. Der SessionKey ist durch den Krankenversicherungsträger zu signieren.

Die Signatur des SessionKeys gewährleistet die Integrität und die Authentizität des übertragenen temporären SessionKeys.

Die Signatur wird über die Base64 kodierten Daten des SessionKeys gebildet. Bei der Signatur handelt es sich um eine nonQES - Signatur (nicht qualifizier te elektronische Signatur). Hier wird bei der Signaturbildung der Content nicht mit eingebettet!

Signaturbildung des SessionKeys

Für beispielsweise ein SessionKey mit diesen Daten:

```
{
  "X": "N4zdP1pH2s9bQocqL1SJntEhJAhx4AA+BdJ5Qa97KJk=",
  "Y": "XYwXh/BoqVExG9KzeGKGEJemlUotVCgA68hNLIIejUc="
}
```

entsteht folgendes Ergebnis:

```
<Wert von X> + | + <Wert von Y>

Hexadezimale Darstellung
37 8C DD 3F 5A 47 DA CF 5B 42 87 2A 2F 54 89 9E D1 21 24 08 71 E0 00 3E 05 D2 79 41 AF 7B 28 99 7C 5D 8C 17 87
F0 68 A9 51 31 1B D2 B3 78 62 86 10 97 A6 95 4A 2D 54 28 00 EB C8 4D 2C 82 1E 8D 47
```

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

Signierung der Ausgangsdaten (VST -> KVT)

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

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

Erweiterte Informationen 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 zu prüfen.

Der öffentliche Schlüssel der Vertrauensstelle IRD für die Prüfung von Signaturen wird innerhalb der TI für die Krankenversicherungsträger an einem gesonderten Endpunkt zum regelmäßigen Download bereitgestellt. Siehe hierzu Downloadpunkt für den Abruf von Schlüsselmaterial der Vertrauensstelle.

Verschlüsselung von Daten

Transportverschlüsselung

Der Schutz der Datenübertragung (Transportkanal) durch die Krankenversicherungsträger an die Vertrauensstelle IRD muss per TLS gesichert sein. Es ist mindestens die Version TLSv1.2 zu verwenden. Im Allgemeinen sind die Vorgaben der Technischen Richtlinie TR-02102-2 "Kryptographische Verfahren: Empfehlungen und Schlüssellängen Teil 2 – Verwendung von Transport Layer Security (TLS)" des Bundesamt für Sicherheit in der Informationstechnik anzuwenden.

Aus Interoperabilitätsgründen sind die Endpunkte der VST mittels eines TLS-Zertifikats gesichert, deren Zertifizierungsstelle in der Browser CA enthalten ist.

Inhaltsverschlüsselung

Wegen des Schutzbedarfs "Hoch" müssen alle patienten- und/oder fallidentifizierenden Daten vor der Übertragung für die Vertrauensstelle auf Feldebene verschlüsselt werden (Ende-zu-Ende-Verschlüsselung). Aus Perspektive der VST sind zwei Übermittlungsrichtungen zu betrachten. Einmal ist es die Verschlüsslung von **Eingangsdaten** durch den KVT bei regelmäßigen Vitalstatusmeldungen (zweimal im Jahr), die laufende Übermittlung von Sterbefällen oder der Mitteilung über Wechsel des Krankenversicherungsträgers eines Versicherten. Zum anderen sind es **Ausgangsdaten** in Form einer Aufforderung durch das Register zur Übermittlung des aktuell bekannten Vitalstatus eines Versicherten (anlassbezogene Vitalstatusmeldung) oder die Bereitstellung einer Liste von Versicherten, für die die Meldepflicht für die KVT in Zukunft entfällt (sog. Anonymisierung).

Verschlüsselung der Eingangsdaten (KVT -> VST)

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

Die öffentlichen Schlüssel der Vertrauensstelle IRD für die Verschlüsselung werden innerhalb der TI für die Krankenversicherungsträger an gesonderten Endpunkten zum regelmäßigen Download bereitgestellt. Siehe hierzu Downloadpunkt für den Abruf von Schlüsselmaterial der Vertrauensstelle.

Verschlüsselung der Ausgangsdaten (VST -> KVT)

Hier wird dasselbe Verfahren wie für die Verschlüsselung der Eingangsdaten verwendet. Bei der Abfrage muss der KVT einen temporär erzeugten öffentlichen Schlüssel (SessionKey) der VST bereitstellen. Siehe Übergreifende Beispiele.

Verwendete Parameter zur Verschlüsselung

Der KVT kennt das Verschlüsselungszertifikat der VST (s. Bereitstellung x.509 Zertifikate der VST aus der Komponenten-PKI der TI). Zunächst prüft der KVT das Zertifikat. Dies ist laut Vorgaben der gematik verpflichtend vor der Verwendung der öffentlichen Schlüssel des Zertifikats durchzuführen. Anschließend verschlüsselt er den eindeutigen Identifikator des Versicherten bzw. zu verschlüsselnden Wert (s. Allgemeine Informationen) mittels des ECIES-Verfahrens [SEC1-2009], [TR-03116-1] und des öffentlichen Schlüssels aus dem geprüften Zertifikat. Dabei verwendet er folgende Verschlüsselungsparameter:

• Der ECDH-Schlüsselaustausch innerhalb von ECIES muss nach [NIST-800-56-A#5.7.1.2] (Hinweis: ist fachlich identisch zu [SEC1-2009#3.3.1]) durchgeführt werden.

• Aus dem gemeinsamen ECDH-Geheimnis muss mit der HKDF nach [RFC-5869] auf Basis von SHA-256 ein AES-256-Bit-Schlüssel abgeleitet werden.

• Dieser Schlüssel muss zur Verschlüsselung des Wertes mittels AES/GCM [NIST-SP-800-38D] (zufällig erzeugter 96-Bit langer IV, 128 Bit langer Authentication-Tag) gebildet werden.

Der konkrete Aufbau des Datenpakets ist unter Übergreifende Beispiele ersichtlich.

Erweiterte HTTP-Response-Header der Vertrauensstelle

Alle HTTP-Programmierschnittstellen innerhalb dieser Spezifikation geben bei der HTTP-Antwort nach der Verarbeitung das jeweils definierte Payload-Fragment, sowie einige Standard-HTTP-Header zurück.

Implizit ist das 9-stellige Institutionskennzeichen (IK) des jeweiligen Krankenversicherungsträgers über die Anmeldung und den Verbindungskontext bei allen HTTP-Programmierschnittstellen **immer** definiert.

Damit die jeweiligen Rückgabe-Payload-Fragmente dennoch explizit mit der anfragenden IK in Verbindung stehen, wird ein zusätzlicher HTTP-Response-Header mit dem Namen 'IK' mitgeliefert. Dieser Header kann bei der Verarbeitung der HTTP-Antwort ignoriert werden, wenn diese Angabe auf Seiten der sendenden Instanz nicht benötigt wird.

Eigenschaften des Headers 'IK'

- der Wert des Headers wird aus dem im Request übergebenen Anmeldetoken extrahiert und unverändert wieder zurückgegeben.
- der Eintrag wird bei allen HTTP-Statuscodes außer 401 mitgeliefert
    - Grund für die Ausnahme ist die Abwesenheit bzw. die Nichtinterpretierbarkeit des Anmeldetokens

Mögliches Beispiel für Antwortkopfzeilen

HTTP/1.1 200 OK
Content-Type: application/json
...

IK: 104127692
Content-Length: 2640

HTTP-Programmierschnittstellen (APIs)

Allgemeiner Hinweis

⚠ Innerhalb der Eigenschaften "IdDatenlieferung" und "IdDatensatz" dürfen niemals 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 Deutschland 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 zur API stehen unterhalb der URL https://xml.ir-d.de/rst/download/vst/krankenversicherungstraeger/ zum Download bereit.

Eigenschaft "IdDatenlieferung"

- definiert die für den meldenden Krankenversicherungsträger eindeutige Id der Datenlieferung
- diese wird durch den Krankenversicherungsträger festgelegt
- Optional: Nein
- Typ: String
- minLength: 3
- maxLength: 40
- 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 Krankenversicherungsträger festgelegt
    - Optional: Nein
    - Typ: String
    - minLength: 3
    - maxLength: 40
    - Schutz: Der Wert wird in Plaintext übertragen
  - IdVersicherter
    - definiert den eindeutigen Identifizierer des Versicherten. Für gesetzlich versicherte Patienten ist dies die eGK-Nummer (unveränderbare Teil der Krankenversichertennummer, KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Für die Heilfürsorge Bundeswehr ist diese Angabe die unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer im Format der Steueridentifikationsnummer entsprechend Identifikationsnummerngesetz (IDNrG).
    - Optional: Nein
    - Format: Byte
    - Typ: String
    - Schutz: Als personenidentifizierendes Merkmal muss dieser Wert für die Vertrauensstelle verschlüsselt werden.
    - Prüfungen:
      - bei Angabe der Krankenversichertennummer:
        - es wird geprüft, dass die Angabe einer gültigen KVNR entspricht (Regex: ^[A-Z]{1}[0-9]{9}$, sowie Prüfung der Prüfsumme)
        - es wird geprüft, dass diese KVNR im jeweiligen Kontext benutzt werden darf ([2])
      - bei Angabe einer Identifikationsnummer der Heilfürsorge Bundeswehr
        - es wird geprüft, ob die Angabe dem festgelegten Schema einer Steueridentifikationsnummer entspricht (Regex: ^[1-9][0-9]{10}$, sowie Prüfung der Prüfsumme)
        - es wird geprüft, dass diese Nummer im jeweiligen Kontext benutzt werden darf ([2])
  - Todesdatum
    - Todesdatum einer Person, die ein meldepflichtiges Implantat trägt. Das Format in unverschlüsselter Form: YYYY-MM-DD (vierstellige Jahreszahl, zweistellige Monatszahl und zweistellige Tageszahl getrennt durch "-". Dies entspricht dem XML-Schema eines Datums, siehe https://www.w3schools.com/XML/schema_dtypes_date.asp . Die Angabe der Zeitzone ist nicht notwendig.
    - 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 muss gesetzt sein, da ansonsten indirekt auf den Vitalstatus des Eintrags geschlossen werden könnte.
    - 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-Statuscode 200 zurückgegeben. Der Content der Antwort ist dabei leer.

Fehlerfälle

Fehler, die bei der Annahme oder Prüfung eines an die API der Vertrauensstelle IRD gesendeten Datenpaketes auftreten, werden auf die gesamte Meldung ausgewertet. Auftretende Fehler werden dem meldenden Krankenversicherungsträger 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 Krankenversicherungsträger asynchron über einen gesonderten Abruf der Daten zur Verfügung gestellt. Die Abfrage von Fehlern wird durch einen Polling-Mechanismus des meldenden Krankenversicherungsträgers erreicht. Wenn zu Datensätzen keine Fehlermeldungen zum Abruf zur Verfügung stehen, so bedeutet es für den Krankenversicherungsträger, dass keine Fehler 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 zurückgemeldet.
Liste der Fehlerfälle:

- HTTP-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.
- HTTP-Statuscode 401: Dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: Dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war. Dies beinhaltet auch die Verwendung von produktiven Daten in der Referenz-Umgebung.
- HTTP-Statuscode 415: Dieser Fehler wird zurückgemeldet, wenn ein falscher Content-Type gesendet wurde.
- HTTP-Statuscode 500: Dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Request


```
{
  "IdDatenlieferung": "2024-01",
  "Meldungen": [
    {
      "IdDatensatz": "8-0000001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlcyBUb2Rlc2RhdHVt"
    },
    {
      "IdDatensatz": "8-0000002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVzIFRvZGVzZGF0dW0="
    }
  ],
  "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 beispielsweise ein JSON-Fragment mit diesen Daten:

```
{
  "IdDatenlieferung": "2024-01",
  "Meldungen": [
    {
      "IdDatensatz": "8-0000001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "Vitalstatus": "SWNoIGJpbiBkZXIgdmVyc2NobMO8c3NlbHRlIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlcyBUb2Rlc2RhdHVt"
    },
    {
      "IdDatensatz": "8-0000002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIElkVmVyc2ljaGVydGVu",
      "Vitalstatus": "SWNoIGJpbiBlaW4gYW5kZXJlciB2ZXJzY2hsw7xzc2VsdGVyIFdlcnQgZGVzIFZpdGFsc3RhdHVz",
      "Todesdatum": "SWNoIGJpbiBlaW4gYW5kZXJlcyB2ZXJzY2hsw7xzc2VsdGVzIFRvZGVzZGF0dW0"
    }
  ]
}
```

entsteht folgendes Ergebnis:

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

Hexadezimale Darstellung
32 30 32 34 2D 30 31 7C 38 2D 30 30 30 30 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C
FC 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 FC 73 73 65 6C 74 65 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 76 65 72 73 63 68 6C FC 73 73 65 6C 74 65 73 20 54 6F 64 65 73 64 61 74 75 6D 7C
38 2D 30 30 30 30 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
FC 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 FC 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 FC 73 73 65 6C 74 65 73 20 54 6F 64 65 73 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

POST /notify/api/v1/vitalstatusnotification/requests

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

Eingabedaten:

Eigenschaft "SessionKey"

- temporärer ECC-PublicKey, der für die Verschlüsselung von Daten an den Versicherungsträger verwendet werden soll
- Optional: Nein
- Felder innerhalb dieses Datenfeldes:
  - X
    - stellt die X-Koordinate des ephemeren öffentlichen Sender-ECC-Schlüssel dar
    - Optional: Nein
    - Format: Byte
    - Typ: String
    - Schutz: Der Wert wird in Klartext übertragen
  - Y
    - stellt die Y-Koordinate des ephemeren öffentlichen Sender-ECC-Schlüssel dar
    - Optional: Nein
    - Format: Byte
    - Typ: String
    - Schutz: Der Wert wird in Klartext übertragen
  - Eigenschaft "Signatur"
    - definiert die Signatur über die Daten des SessionKeys
    - Optional: Nein

- Format: Byte
- Typ: String

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der HTTP-Statuscode 204 zurückgegeben. Der Content ist dabei leer.

Wenn Daten zum Abruf vorhanden sind, wird der HTTP-Statuscode 200 zurückgegeben. Als Content der Antwort wird die Liste der Versicherten zurückgegeben, zu denen eine Vitalstatusmeldung angefragt wird. Die Antwort ist vom Typ "RequestsForVitalStatusNotificationBundle".

Eigenschaft "Anfragen"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "RequestForVitalStatusNotification"
- Felder innerhalb eines Datensatzes:
  - IdVersicherter
    - definiert den eindeutigen Identifizierer des Versicherten. Für gesetzlich versicherte Patienten ist dies die eGK-Nummer (unveränderbare Teil der Krankenversichertennummer, KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Für die Heilfürsorge Bundeswehr ist diese Angabe die unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer im Format der Steueridentifikationsnummer entsprechend Identifikationsnummerngesetz (IDNrG).
    - Optional: Nein
    - Format: Byte
    - Typ: String
    - Schutz: Als personenidentifizierendes Merkmal ist dieser Wert für den Krankenversicherungsträger mit dem für diese Übertragung erzeugten Sitzungsschlüssel (siehe Verschlüsselung der Ausgangsdaten) 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 zurückgemeldet.
Liste der Fehlerfälle:

- HTTP-Statuscode 401: dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- HTTP-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 des SessionKeys

Siehe Beschreibung dazu.

Signierung der Ausgangsdaten

Die Antworten, 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 beispielsweise ein 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 FC 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 FC 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.

Verschlüsselung der Ausgangsdaten

Die Daten, die an den Versicherungsträger zurückgesendet werden, sind durch die Vertrauensstelle gegen den temporären Schlüssel "SessionKey" verschlüsselt worden.

Weitere Informationen finden sich unter dem Punkt Verschlüsselung von Ausgangsdaten.

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

POST /notify/api/v1/anonymizationnotifications

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

Eingabedaten:

Eigenschaft "SessionKey"

- temporärer ECC-PublicKey, der für die Verschlüsselung von Daten an den Versicherungsträger verwendet werden soll
- Optional: Nein
- Felder innerhalb dieses Datenfeldes:
    - X
        - stellt die X-Koordinate des ephemeren öffentlichen Sender-ECC-Schlüssel dar
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Der Wert wird in Klartext übertragen
    - Y
        - stellt die Y-Koordinate des ephemeren öffentlichen Sender-ECC-Schlüssel dar
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Der Wert wird in Klartext übertragen
    - Eigenschaft "Signatur"
        - definiert die Signatur über die Daten des SessionKeys
        - Optional: Nein
        - Format: Byte

- Typ: String

Rückgabe

Wenn keine Daten zum Abruf vorhanden sind, wird der HTTP-Statuscode 204 zurückgegeben. Der Content ist dabei leer.

Wenn Daten zum Abruf vorhanden sind, wird der HTTP-Statuscode 200 zurückgegeben. Als Content der Antwort wird die Liste der Versicherten zurückgegeben, zu denen eine Anonymisierung innerhalb der Vertrauensstelle und der Registerstelle vorgenommen wurde. Die Antwort ist 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 eindeutigen Identifizierer des Versicherten. Für gesetzlich versicherte Patienten ist dies die eGK-Nummer (unveränderbare Teil der Krankenversichertennummer, KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Für die Heilfürsorge Bundeswehr ist diese Angabe die unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer im Format der Steueridentifikationsnummer entsprechend Identifikationsnummerngesetz (IDNrG).
    - Optional: Nein
    - Format: Byte
    - Typ: String
    - Schutz: Als personenidentifizierendes Merkmal ist dieser Wert für den Krankenversicherungsträger mit dem Sitzungsschlüssel (siehe Verschlüsselung der Ausgangsdaten)
    - 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:

- HTTP-Statuscode 401: dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- HTTP-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 des SessionKeys

Siehe Beschreibung dazu.

Signierung der Ausgangsdaten

Die Antworten, 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 beispielsweise ein 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.

Verschlüsselung der Ausgangsdaten

Die Daten, die an den Versicherungsträger zurückgesendet werden, sind durch die Vertrauensstelle gegen den temporären Schlüssel "SessionKey" verschlüsselt worden.

Weitere Informationen finden sich unter dem Punkt Verschlüsselung von Ausgangsdaten.

API für den Abruf von Verarbeitungsresultaten der asynchronen Verarbeitung von Vitalstatus-Meldungen auf Seiten der Vertrauensstelle oder der Registerstelle

POST /notify/api/v1/vitalstatusnotification/processingresults

Diese Schnittstelle ist für den Abruf von Verarbeitungsresultaten der asynchronen Verarbeitung von Vitalstatus-Meldungen auf Seiten der Vertrauensstelle oder der Registerstelle zu verwenden.

Eingabedaten:

- IdDatenlieferung
  - definiert die für den meldenden Krankenversicherungsträger eindeutige Id der Datenlieferung, zu der die Ergebnisse der Datenverarbeitung abgerufen werden sollen. Diese Id wurde bei der Datenlieferung durch den meldenden Versicherungsträger festgelegt
  - Optional: Nein
  - Typ: String
  - minLength: 3
  - maxLength: 40
  - Schutz: Der Wert wird in Plaintext übertragen

Rückgabe

Wenn keine Daten zu der übergebenen Id der Datenlieferung zum Abruf vorhanden sind, wird der HTTP-Statuscode 204 zurückgegeben. Der Content ist dabei leer. Dies kann entweder bedeuten, dass die Daten zu dieser Datenlieferung noch nicht verarbeitet wurden bzw. die Ergebnisse der Verarbeitung bereits abgerufen und dadurch gelöscht wurden.

Wenn Daten zum Abruf vorhanden sind, wird der HTTP-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. Die Antwort ist vom Typ "VitalStatusNotificationProcessingErrorsBundle". Wenn es zu der angegebenen Datenlieferung keine Fehler gibt (also alle Datensätze innerhalb der Lieferung in Ordnung sind), so wird nur der Statuscode 200 zurückgegeben. Der Content der Antwort ist dabei leer.

Eigenschaft "Fehler"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "VitalStatusNotificationProcessingErrorResult"
- Felder innerhalb eines Datensatzes:
  - IdDatensatz
    - definiert die eindeutige Id des Datensatzes (innerhalb der Datenlieferung)
    - Optional: Nein
    - Typ: String
    - minLength: 3
    - maxLength: 40
    - 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 zurückgemeldet.
Liste der Fehlerfälle:

- HTTP-Statuscode 401: dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- HTTP-Statuscode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response


```
{
        "Fehler":
        [
            {
                        "IdDatensatz": "8-0000001",
                        "Code": "DecryptionError"
            },
            {
                        "IdDatensatz": "8-0000002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```


Signierung der Ausgangsdaten

Die Antworten, 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 beispielsweise ein JSON-Fragment mit diesen Daten:

```
{
        "Fehler":
        [
            {
                        "IdDatensatz": "8-0000001",
                        "Code": "DecryptionError"
            },
            {
                        "IdDatensatz": "8-0000002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ]
}
```

entsteht folgendes Ergebnis:

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

Hexadezimale Darstellung
38 2D 30 30 30 30 30 30 31 7C 44 65 63 72 79 70 74 69 6F 6E 45 72 72 6F 72 7C 38 2D 30 30 30 30 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 OpenAPI_InsuranceUpdateService-API_V1.json unter dem Typ "InsuranceUpdateNotificationBundle" beschrieben. Die Dokumente zur API stehen unterhalb der URL https://xml.ir-d.de/rst/download/vst/krankenversicherungstraeger/ zum Download bereit.

Eigenschaft "IdDatenlieferung"

- definiert die für den meldenden Krankenversicherungsträger eindeutige Id der Datenlieferung
- diese wird durch den Krankenversicherungsträger festgelegt
- Optional: Nein
- Typ: String
- minLength: 3
- maxLength: 40
- 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 Krankenversicherungsträger festgelegt
        - Optional: Nein
        - Typ: String
        - minLength: 3
        - maxLength: 40
        - Schutz: Der Wert wird in Plaintext übertragen
    - IdVersicherter
        - definiert den eindeutigen Identifizierer des Versicherten bei dem meldenden Versicherungsträger. Für gesetzlich versicherte Patienten ist dies die eGK-Nummer (unveränderbare Teil der Krankenversichertennummer, KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V). Für die Heilfürsorge Bundeswehr ist diese Angabe die unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer im Format der Steueridentifikationsnummer entsprechend Identifikationsnummerngesetz (IDNrG).
        - Optional: Nein

        - Format: Byte
        - Typ: String
        - Schutz: Der Wert muss für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - bei Angabe der 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 diese KVNR im jeweiligen Kontext benutzt werden darf ([2])
            - bei Angabe einer Identifikationsnummer der Heilfürsorge Bundeswehr:
                - es wird geprüft, ob die Angabe dem festgelegten Schema einer Steueridentifikationsnummer entspricht (Regex: ^[1-9][0-9]{10}$, sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass diese Nummer im jeweiligen Kontext benutzt werden darf ([2])
    - IdVersicherterNeu
        - definiert den aktuellen, eindeutigen Identifizierer des Versicherten beim Versicherungsträger. Dieser Identifizierer ist der unveränderbare Teil der Krankenversichertennummer (KVNR) nach § 290 des Fünften Buches Sozialgesetzbuch (SGB V) bzw. eine andere eindeutige, unveränderbare und nach einheitlichen Kriterien gebildete Identifikationsnummer entsprechend §17 Abs. 4 IRegG.
          Bei Beendigung eines Versicherungsverhältnisses ohne Folgeversicherung muss dieses Feld mit dem Wert "unbekannt" befüllt sein (Wert vor der Verschlüsselung).
          Bei Beginn eines Versicherungsverhältnisses ohne Vorversicherung muss dieser Wert gleich der IdVersicherter sein.
        - Optional: Nein
        - Format: Byte
        - Typ: String
        - Schutz: Der Wert muss für die Vertrauensstelle verschlüsselt werden
        - Prüfungen:
            - bei Angabe der Krankenversichertennummer:
                - es wird geprüft, dass die Angabe einer gültigen KVNR entspricht (Regex: ^[A-Z]{1}[0-9]{9}$ sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass diese KVNR im jeweiligen Kontext benutzt werden darf ([2])
            - bei Angabe einer Identifikationsnummer der Heilfürsorge Bundeswehr:
                - es wird geprüft, ob die Angabe dem festgelegten Schema einer Steueridentifikationsnummer entspricht (Regex: ^[1-9][0-9]{10}$, sowie Prüfung der Prüfsumme)
                - es wird geprüft, dass diese Nummer im jeweiligen Kontext benutzt werden darf ([2])
    - IkNeu
        - definiert das Haupt-IK (Institutionskennzeichen) des aktuellen Versicherungsträgers des Versicherten.
          Bei Beendigung eines Versicherungsverhältnisses ohne Folgeversicherung muss dieses Feld mit dem Wert "unbekannt" befüllt sein.
          Bei Beginn eines Versicherungsverhältnisses ohne Vorversicherung muss dieser Wert gleich des IkMelder sein (IkMelder ist die IK, die bei der der Anmeldung angegeben ist).
        - Optional: Nein
        - Typ: String
        - Schutz: Der Wert wird in Plaintext übertragen
        - Prüfungen:
            - es wird geprüft, dass die Angabe einem validen Institutionskennzeichen entspricht (Regex: ^[0-9]{9}$ 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 HTTP-Statuscode 200 zurückgegeben.

Fehlerfälle

Fehler, die bei der Annahme oder Prüfung eines Datenpaketes an die API der Vertrauensstelle IRD auftreten, werden auf die gesamte Meldung ausgewertet. Auftretende Fehler werden dem meldenden Krankenversicherungsträger 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 Krankenversicherungsträger asynchron über einen gesonderten Abruf der Daten zur Verfügung gestellt. Die Abfrage von Fehlern wird durch einen Polling-Mechanismus des meldenden Krankenversicherungsträgers erreicht. Wenn zu Datensätzen keine Fehlermeldungen zum Abruf zur Verfügung stehen, so bedeutet es für den Krankenversicherungsträger, dass keine Fehler 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 zurückgemeldet.
Liste der Fehlerfälle:

- HTTP-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.
- HTTP-Statuscode 401: dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war. Dies beinhaltet auch die Verwendung von produktiven Daten in der Referenz-Umgebung.
- HTTP-Statuscode 415: dieser Fehler wird zurückgemeldet, wenn ein falscher Content-Type gesendet wurde.

- HTTP-Statuscode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Request


```
{
  "IdDatenlieferung": "2024-001",
  "Meldungen": [
    {
      "IdDatensatz": "8-0000001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "IdVersicherterNeu": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlck5ldQ==",
      "IkNeu": "260326823"
    },
    {
      "IdDatensatz": "8-0000002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVy",
      "IdVersicherterNeu":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVyTmV1",
      "IkNeu": "unbekannt"
    }
  ],
  "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```


Signaturbildung der Eingangsdaten

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

Für beispielsweise ein JSON-Fragment mit diesen Daten:


```
{
  "IdDatenlieferung": "2024-001",
  "Meldungen": [
    {
      "IdDatensatz": "8-0000001",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlcg==",
      "IdVersicherterNeu": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyBJZFZlcnNpY2hlcnRlck5ldQ==",
      "IkNeu": "260326823"
    },
    {
      "IdDatensatz": "8-0000002",
      "IdVersicherter": "SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVy",
      "IdVersicherterNeu":
"SWNoIGJpbiBlaW4gdmVyc2NobMO8c3NlbHRlciBXZXJ0IGRlcyB6d2VpdGVuIElkVmVyc2ljaGVydGVyTmV1",
      "IkNeu": "unbekannt"
    }
  ]
}
```


entsteht folgendes Ergebnis:

```
<Wert der Eigenschaft IdDatenlieferung> + | + <Wert des IdVersicherten Datensatz 1> + | + <Wert des
IdVersicherterNeu Datensatz 1> + <Wert des IkNeu Datensatz 1> + | + <Wert des IdVersicherten Datensatz 2> + | +
<Wert des IdVersicherterNeu Datensatz 2> + |+ <Wert des IkNeu Datensatz 2>

Hexadezimale Darstellung
38 2D 30 30 30 30 30 30 31 7C 49 63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C FC 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 63 68 6C FC 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 4E 65 75 7C 32 36 30 33 32 36 38 32 32 7C 32 36 30 33 32 36 38 32 33 7C 38 2D 30 30 30 30 30 30 32 7C 49
63 68 20 62 69 6E 20 65 69 6E 20 76 65 72 73 63 68 6C FC 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 FC 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 4E 65 75 7C 75 6E 62 65 6B 61 6E 6E 74
```

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

API für den Abruf von Verarbeitungsresultaten der asynchronen Verarbeitung von Meldungen bzgl. dem Versicherungswechsel von im Implantateregister erfassten Versicherten auf Seiten der Vertrauensstelle

POST /notify/api/v1/insuranceupdatenotification/processingresults

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

Eingabedaten:

- IdDatenlieferung
  - definiert die für den meldenden Krankenversicherungsträger eindeutige Id der Datenlieferung, zu der die Ergebnisse der Datenverarbeitung abgerufen werden sollen
  - Optional: Nein
  - Typ: String
  - minLength: 3
  - maxLength: 40
  - Schutz: Der Wert wird in Plaintext übertragen

Rückgabe

Wenn keine Daten zu der übergebenen Id der Datenlieferung zum Abruf vorhanden sind, wird der HTTP-Statuscode 204 zurückgegeben. Der Content ist dabei leer. Dies kann entweder bedeuten, dass die Daten zu dieser Datenlieferung noch nicht verarbeitet wurden bzw. die Ergebnisse der Verarbeitung bereits abgerufen und dadurch gelöscht wurden.

Wenn Daten zum Abruf vorhanden sind, wird der HTTP-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 Vertrauensstelle zu Fehlern kam. Die Antwort ist vom Typ "InsuranceUpdateProcessingErrorsBundle". Wenn es zu der angegebenen Datenlieferung keine Fehler gibt (also alle Datensätze innerhalb der Lieferung in Ordnung sind), so wird nur der Statuscode 200 zurückgegeben. Der Content der Antwort ist dabei leer.

Eigenschaft "Fehler"

- Definiert die Datensätze innerhalb dieser Datenlieferung
- Optional: Nein
- Typ: Array
- Typ der Elemente innerhalb des Arrays: "InsuranceUpdateProcessingErrorResult"
- Felder innerhalb eines Datensatzes:
  - IdDatensatz
    - definiert die eindeutige Id des Datensatzes (innerhalb der Datenlieferung)
    - Optional: Nein
    - Typ: String
    - minLength: 3
    - maxLength: 40
    - 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 zurückgemeldet.
Liste der Fehlerfälle:

- HTTP-Statuscode 401: dieser Fehler wird zurückgemeldet, wenn kein Authentisierungstoken vorhanden ist (s. Authentisierung und Authentifizierung) oder die Authentifizierung an der Schnittstelle fehlgeschlagen ist.
- HTTP-Statuscode 403: dieser Fehler wird zurückgemeldet, wenn die Anfrage nicht autorisiert war.
- HTTP-Statuscode 500: dieser Fehler wird zurückgemeldet, wenn ein interner Fehler bei der Verarbeitung aufgetreten ist.

Beispiel Response

```
{
        "Fehler":
        [
            {
                        "IdDatensatz": "8-0000001",
                        "Code": "DecryptionError"
            },
            {
                        "IdDatensatz": "8-0000002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ],
        "Signatur":
"SWNoIGJpbiBkaWUgU2lnbmF0dXIgw7xiZXIgZGllIGdlc2FtdGVuIFZpdGFsc3RhdHVzLU1lbGR1bmdlbiB1bmQgZGllIElkTWVsZHVuZw=="
}
```

Signierung der Ausgangsdaten

Die Antworten, 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 beispielsweise ein JSON-Fragment mit diesen Daten:

```
{
        "Fehler":
        [
            {
                        "IdDatensatz": "8-0000001",
                        "Code": "DecryptionError"
            },
            {
                        "IdDatensatz": "8-0000002",
                        "Code": "WrongFormatIdVersicherter"
              }
        ]
}
```

entsteht folgendes Ergebnis:

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

Hexadezimale Darstellung
38 2D 30 30 30 30 30 30 31 7C 44 65 63 72 79 70 74 69 6F 6E 45 72 72 6F 72 7C 38 2D 30 30 30 30 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 einen Krankenversicherungsträger gesendet werden, entschlüsselt werden können sowie die Signatur eines Datenpaketes geprüft werden kann.

Verschlüsselung der Eingangsdaten (KVT -> VST)

Bei der Verschlüsselung der Eingangsdaten wird sich an dem Konzept der gematik des E-Rezeptes angelehnt. Eine Beispiel-Implementierung zur Erstellung der verschlüsselten Datenfelder findet sich unter der 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). In der

Bei der Verschlüsselung wird dabei für jede Datenübertragung ein eigenes temporäres ECC-Schlüsselpaar gebildet. Dieses Schlüsselpaar dient dann als Eingangsparameter des ECDHBasicAgreement, mit dessen Hilfe gegen das öffentliche Zertifikat der Vertrauensstelle für jeden zu verschlüsselnden Wert ein temporäres SharedSecret gebildet wird. Dieses temporäre gemeinsame Geheimnis (shared secret) 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-256-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 temporären 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). Siehe auch Struktur des verschlüsselten Datenfelds:

```
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 Implementierung "1".
Die nächsten 32 Bytes definieren die X-Koordinate des temporäres gebildeten Schlüsselpaares.
Die nächsten 32 Bytes definieren die Y-Koordinate des temporäres 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 jede Datenübertragung muss ein eigenes, temporäres Schlüsselpaar gebildet und verwendet werden. Nach der Datenübertragung kann dieses Schlüsselpaar wieder verworfen werden.
⚠️ Für jede AES-GCM-Verschüsselungsoperation muss ein eigener, zufällig gewürfelter IV gebildet werden.

Verschlüsselung der Ausgangsdaten (VST -> KVT)

Bei Operationen, in denen Daten durch die Vertrauensstelle an den Krankenversicherungsträger (KVT) verschlüsselt übertragen werden müssen, muss der KVT beim Aufruf der API-Funktion einen temporären Schlüssel ("SessionKey") mitsenden.

Dazu muss der aufrufende KVT ein für diesen Datenabruf temporäres ECC-Schlüsselpaar generieren. Der öffentliche Schlüssel entspricht dem SessionKey (die x- und y-Koordinaten des öffentlichen Schlüssels aus dem temporären ECC-Schlüsselpaar), der signiert an die Vertrauensstelle übermittelt werden muss. Das temporäre Schlüsselpaar wird dabei äquivalent wie bei der Verschlüsselung von Eingangsdaten gebildet (siehe Beispiel oben). Die VST kann aus dem SessionKey ein gemeinsames Geheimnis ableiten, um damit für den KVT das Ergebnis zu verschlüsseln. Dieses Schlüsselpaar muss sich den gesamten Datenabruf zur Entschlüsselung der empfangenen Daten gemerkt werden. Nach der Entschlüsselung ist das Schlüsselpaar zu verwerfen.

Die Parameter finden sich unter Parameter für die Verschlüsselung bzw. unten unter Struktur des erzeugten verschlüsselten Datenfeldes.

Die Struktur der von der VST gelieferten Ausgangdaten lässt sich unten finden, siehe Struktur Verschlüsseltes Datenfeld.

Struktur des erzeugten verschlüsselten Datenfeldes

Dem erzeugten Chiffrat wird ein Byte als Format-ID (aktuell 0x01) des Chiffrats vorangestellt. Damit entsteht folgende Struktur unter der Maßgabe, dass der öffentliche Schlüssel ein Punkt auf der Kurve brainpoolP256r1 [RFC-5639] ist:

Offset	Erläuterung
0	Format-ID (0x01)
1	X-Wert des ephemeren öffentlichen Sender-Schlüssels
33	Y-Wert des ephemeren öffentlichen Sender-Schlüssels
65	IV (12 Byte)
77	Chiffrat des Wertes (Länge gleich; len(v))
77 + len(v)	AES/GCM Authentication Tag (16 Byte)

Diese Byte-Folge wird mittels Base64-kodiert und stellt das Datenfeld dar.

Erläuterung: Bei jeder Datenlieferung wird vom KVT zufällig ein ephemeres Sender-Schlüsselpaar erzeugt, dass zur Verschlüsselung der Werte verwendet wird. Damit wird ein einseitig authentisierter ECDH-Schlüsselaustausch durchgeführt (öffentlicher Schlüssel VST). Damit entsteht jeweils ein neuer AES-Schlüssel, womit das Chiffrat mit jeder Erzeugung unterschiedlich.

Initialisierungsvektor (IV) muss für jede Verschlüsselungsoperation zufällig gewählt werden

Für jede Verschüsselungsoperation mit dem so erzeugten symmetrischen Schlüssel muss für AES-GCM ein eigener, zufällig gewürfelter IV gebildet werden.

Durch dieses randomiserte Verfahren wird erreicht, dass die eindeutigen Identifikatoren der Versicherten jeweils unterschiedlich chiffriert werden. So sind mehrere Vitalstatusmeldungen (halbjährliche Pflichtmeldung oder anlassbezogene Aufforderung zur Übermittlung durch das Register), Übermittlung von Sterbefällen, bzw. Kassenwechsel, oder die Aufforderung zur Anonymisierung, die ein und desselben Versicherten betreffen, nicht ersichtlich.

Signierung der Anmeldedaten (KVT -> VST)

Die Signierung von Daten mit dem Schlüsselmaterial der TI ist nur mithilfe einer Aktion über den Konnektor bzw. des Basis-/KTR-Consumers möglich. Dazu wird ein entsprechender Endpunkt des Konnektors bzw. des Basis-/KTR-Consumers angesprochen, bei dem dann die aufgerufene SOAP-Action 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	<pre><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='beliebige Zeichenkette'>
	                <ns4:OptionalInputs>
	                    <dss:SignatureType>urn:ietf:rfc:5652</dss:SignatureType>
	                    <ns4:IncludeEContent>true</ns4:IncludeEContent>
	                </ns4:OptionalInputs>
	                <ns4:Document ShortText='beliebige Zeichenkette' ID='beliebige Zeichenkette'>
	                    <dss:Base64Data>VGVzdAo=</dss:Base64Data>
	                </ns4:Document>
	                <ns4:IncludeRevocationInfo>true</ns4:IncludeRevocationInfo>
	            </ns4:SignRequest>
	        </ns4:SignDocument>
	    </soap:Body>
	</soap:Envelope></pre>

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 in dem Datenpaket eingebettet wird, muss `<ns4:IncludeEContent>true</ns4:IncludeEContent>` gesetzt sein.
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.

Signierung der Eingangsdaten (payload) (KVT -> VST)

Die Signierung von Daten mit dem Schlüsselmaterial der TI ist nur mithilfe einer Aktion über den Konnektor bzw. des Basis-/KTR-Consumers möglich. Dazu wird ein entsprechender Endpunkt des Konnektors bzw. des Basis-/KTR-Consumers angesprochen, bei dem dann die aufgerufene SOAP-Action 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='beliebige Zeichenkette'>`
	`<ns4:OptionalInputs>`
	`<dss:SignatureType>urn:ietf:rfc:5652</dss:SignatureType>`
	`<ns4:IncludeEContent>false</ns4:IncludeEContent>`
	`</ns4:OptionalInputs>`
	`<ns4:Document ShortText='beliebige Zeichenkette' ID='beliebige Zeichenkette'>`
	`<dss:Base64Data>VGVzdAo=</dss:Base64Data>`
	`</ns4:Document>`
	`<ns4:IncludeRevocationInfo>true</ns4:IncludeRevocationInfo>`
	`</ns4:SignRequest>`
	`</ns4:SignDocument>`
	`</soap:Body>`
	`</soap:Envelope>`

Die Vorgehensweise ist identisch zu Signierung der Anmeldedaten, hier wird jedoch der Content nicht eingebettet, also

`<ns4:IncludeEContent>false</ns4:IncludeEContent> gesetzt.`

Signierung der Ausgangsdaten (VST -> KVT)

Signaturen von Datenpaketen, die durch die Vertrauensstelle gebildet werden, werden durch ECDSA-Signaturen abgebildet und an den empfangenden KVT gesendet. Das öffentliche Signaturzertifikat der Vertrauensstelle lässt sich über den gesonderten öffentlichen Downloadpunkt abrufen.

Die Signaturen werden über den SHA256-Hash der Nachricht gebildet.

Mithilfe des abgerufenen Signaturzertifikats lässt sich dann die Signatur validieren, die durch die Vertrauensstelle gebildet wurde.

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 Versionen der Services lassen sich mithilfe der Herstellerinformationen bestimmen.

Die gematik-Spezifikation mit erweiterten Informationen lässt sich unter den Dokumenten "gemSpec_Basis_KTR_Consumer" bzw. "gemSpec_Kon" und der "gemSpec_Krypt" auf 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 Krankenversicherungsträgers angepasst werden.
- Das Handle der zu verwendenden Karte in `<ns2:CardHandle></ns2:CardHandle>` referenziert auf die zu verwendende 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>

Nichtfunktionale Festlegungen

aktuell keine

Test

Definition von Testdaten

Anwendungstests sind entsprechend der Vorgaben der gematik nur in der dafür bereitgestellten Referenzumgebung (RU) der TI erlaubt. Die Produktivumgebung (PU) der TI darf ausschließlich mit produktiven Daten genutzt werden.

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.

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.

Testdaten Identifikationsnummer der Heilfürsorge BW

Für die Identifikationsnummer der Heilfürsorge BW stehen aktuell die Formatinformationen zur Identifikationsnummer zur Verfügung. Als Testnummer ist auf Seite 7 die TestID *02476291358* angegeben.

Regeln für die TI-Umgebungen

Referenzumgebung der TI: In der RU dürfen für das patienten-identifizierende Datum *IdVersicherter* nur Werte aus den oben beschriebenen Testdaten verwendet werden. Die Schnittstelle lehnt bei Nichtbeachtung dieser Regel die Annahme der Daten ab und gibt einen Fehlercode an den Sender zurück.

Produktivumgebung der TI: In der PU werden für das patienten-identifizierende Datum *IdVersicherter* produktive, d.h. einem Patienten zugeordnete Daten erwartet.
Aufrufe mit Werten für *IdVersicherter* aus den oben beschriebenen Testdaten werden aktuell als Ausnahme ausschließlich zum Test der korrekten Anbindung des meldenden Krankenversicherungsträgers an die Produktivumgebung der TI akzeptiert und von der Vertrauensstelle nicht verarbeitet.

Kontakt

Robert Koch-Institut
Referat VIG - Vertrauensstellen
Nordufer 20
13353 Berlin

E-Mail: Vertrauensstelle-IRD@rki.de