IQTIG Institut für Qualitätssicherung und Transparenz im Gesundheitswesen

Pseudonymisierungsprogramm (PSP)

Programm zur systemweit einheitlichen Leistungserbringerpseudonymisierung

Version: 4.0.8 / Stand: 16.10.2026

Inhaltsverzeichnis

1 Allgemeines . . . 1
Übersicht . . . 1
Zielgruppe . . . 1
2 Einleitung . . . 2
2.1 Verschlüsselungsverfahren zur Pseudonymisierung . . . 2
2.2 Anwendungsbereich . . . 3
2.3 Eigenschaften der Leistungserbringerpseudonymisierung . . . 4
2.3.1 Institutskennzeichen-/Betriebsstättennummer-Pseudonym . . . 4
2.3.2 Standortpseudonym . . . 4
2.3.3 Lebenslange Arztnummer . . . 4
2.3.4 Durchführung der Pseudonymisierung . . . 5
2.3.5 Durchführung der Depseudonymisierung . . . 5
3 Installation . . . 6
3.1 Systemvoraussetzungen . . . 6
3.2 Programmbestandteile . . . 6
3.3 Installation . . . 6
3.4 Aktualisierung . . . 6
4 Programmaufbau . . . 7
4.1 Konsolenanwendung (CLI) . . . 7
4.1.1 Mapping - Übermittlung der Änderungen von Leistungserbringern an die BAS . . . 10
Die Eingabedatei . . . 10
Ergänzung der Eingabedatei und Übertragung der Ausgabedatei . . . 13
Das Spezifikationsjahr . . . 13
Das Ausgabeverzeichnis . . . 13
Die Zertifikatsdatei . . . 14
4.1.2 Positivliste - Übermittlung aller Leistungserbringerpseudonyme an die BAS . . . 15
Das Spezifikationsjahr . . . 15
Die Eingabedatei . . . 15
Das Ausgabeverzeichnis . . . 16
Die Zertifikatsdatei . . . 16
4.1.3 Pseudonymisierung - Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von Klartext-Kennungen . . . 17
Die Eingabedatei . . . 17
Das Ausgabeverzeichnis . . . 19
Die Zertifikatsdatei . . . 19
Die ausgegebene CSV-Datei . . . 19
4.1.4 Pseudonymisierung - Erstellung einer Liste von IAnr-Pseudonymen aus einer Eingabeliste von IAnr-Kennungen . . . 20
Die Eingabedatei . . . 20
Das Ausgabeverzeichnis . . . 21
Die Zertifikatsdatei . . . 21

Die ausgegebene CSV-Datei . . . 21
4.1.5 Depseudonymisierung - Depseudonymisierung einer Eingabeliste im CSV-Format von Pseudonymen mit Ausgabe als CSV-Datei . . . 22
Die Eingabedatei . . . 22
Das Ausgabeverzeichnis . . . 23
Die Schlüsseldatei . . . 23
Das Passwort . . . 23
Die ausgegebene CSV-Datei . . . 24
4.1.6 Depseudonymisierung - Depseudonymisierung einer Eingabeliste von IAnr-Pseudonymen im CSV-Format mit Ausgabe als CSV-Datei . . . 25
Die Eingabedatei . . . 25
Das Ausgabeverzeichnis . . . 26
Die Schlüsseldatei . . . 26
Das Passwort . . . 26
Die ausgegebene CSV-Datei . . . 26
4.1.7 Depseudonymisierung - Übermittlung von depseudonymisierten Leistungserbringerkennungen an die BAS . . . 28
Die Eingabedatei . . . 28
Das Ausgabeverzeichnis . . . 29
Die Schlüsseldatei . . . 30
Das Passwort . . . 30
4.1.8 Zertifikatswechsel . . . 31
Das Spezifikationsjahr . . . 31
Die Eingabedatei . . . 31
Das Ausgabeverzeichnis . . . 33
Die alte Zertifikatsdatei . . . 33
Die neue Zertifikatsdatei . . . 33
4.1.9 Schlüsselpaarerzeugung . . . 34
Typ der Datenannahmestelle . . . 34
Die Landesangabe . . . 34
Das Von-Datum des Gültigkeitszeitraumes . . . 34
Das Bis-Datum des Gültigkeitszeitraumes . . . 35
Das Passwort . . . 35
Das Ausgabeverzeichnis . . . 35
4.1.10 Rückgabewerte und Fehlerbehandlung . . . 35
4.2 Programmierschnittstelle (API) . . . 36
4.2.1 Interface KeyManagement . . . 36
Erzeugung eines Zertifikates mit privatem Schlüssel . . . 36
Erzeugung eines Zertifikates mit privatem Schlüssel ausgegeben als PKCS12-Datei . . . 37
Extraktion eines Zertifikates aus PKCS12-Datei und Ausgabe in Outputstream . . . 37
Extraktion eines Zertifikates aus PKCS12-Datei . . . 38
Einlesen eines x509-Zertifikates aus Schlüsseldatei im PEM-Format . . . 38
4.2.2 Interface Pseudonymisator . . . 39

Pseudonymisieren einer IK- oder BSNR entsprechend dem übergebenen Pseudonymisierungsverfahren . . . 39
Pseudonymisieren einer StandortId entsprechend dem übergebenen Pseudonymisierungsverfahren . . . 39
Pseudonymisieren einer lebenslangen Arztnummer entsprechend dem übergebenen Pseudonymisierungsverfahren . . . 40
Pseudonymisieren einer Liste von IkBs/StoId-Kennungen . . . 40
Pseudonymisieren einer Liste von IAnr-Kennungen . . . 42
4.2.3 Interface Depseudonymisator . . . 42
Depseudonymisierung eines Pseudonyms . . . 42
Depseudonymisierung einer Liste von IkBs- bzw. StoId-Pseudonymen . . . 43
Depseudonymisierung einer Liste von IAnr-Pseudonymen . . . 44
4.2.4 API einzeln einbinden . . . 45
5 Migration . . . 46
5.1 Migration der API von Version 2.x zu Version 4.x . . . 46
5.1.1 Interface CertificateStore . . . 46
5.1.2 Interface CsvDepseudonymisator . . . 46
5.1.3 Interface CsvPseudonymisator . . . 46
5.1.4 Interface IkbsAndStoDepseudonymisator . . . 47
Methode zur Depseudonymisierung eines Strings . . . 47
Depseudonymisierung einer Liste von Pseudonymen . . . 47
5.1.5 Interface IkbsAndStoPseudonymisator . . . 48
Methode zur Erzeugung eines IkBs-Pseudonyms . . . 48
Methode zur Erzeugung einer Liste von IkBs-Pseudonymen . . . 48
Methode zur Erzeugung eines StoId-Pseudonyms . . . 49
Methode zur Erzeugung einer Liste von StoId-Pseudonymen . . . 49
Methode zur Erzeugung eines IAnr-Pseudonyms . . . 50
Methode zur Erzeugung einer Liste von IAnr-Pseudonymen . . . 50
5.1.6 Interface Registration . . . 51
5.1.7 Interface StringDepseudonymisator . . . 51
5.1.8 Interface StringPseudonymisator . . . 51
5.1.9 Interface X509CertificateGenerator . . . 51
Migrationspfad anhand des Beispiels der Erzeugung eines Zertifikates mit privatem Schlüssel abgelegt in einem PKCS12-Keystore . . . 51
5.2 Migration des CLI von Version 2.x zu Version 4.x . . . 53
5.2.1 Generierung von Zertifikaten . . . 53
5.2.2 Pseudonymisierung von CSV zu CSV . . . 54
5.2.3 Depseudonymisierung von CSV zu CSV . . . 55
5.2.4 Übermittlung der Zusatzdaten an die BAS . . . 56
6 Hilfe und Support . . . 57

1 Allgemeines

Übersicht

Dieses Dokument beschreibt die systemweit einheitliche Leistungserbringerpseudonymisierung sowie die Nutzung des vom IQTIG dafür entwickelten Pseudonymisierungsprogramms (PSP) in der oben angegebenen Version.

Zielgruppe

Das Anwenderhandbuch richtet sich an administrative Mitarbeitende der Datenannahmestellen (DAS) sowie Softwareentwickler und Softwareentwicklerinnen, die mit der Umsetzung der Leistungserbringerpseudonymisierung betraut sind.

2 Einleitung

Die Erhebung und Auswertung von Qualitätssicherungsdaten (QS-Daten) im Rahmen der Richtlinie zur datengestützten einrichtungsübergreifenden Qualitätssicherung (DeQS-RL) [https://www.g-ba.de/richtlinien/105] erfordert eine einheitliche Pseudonymisierung der leistungserbringeridentifizierenden Daten. Für die Zusammenführung der Qualitätssicherungsdaten aus den unterschiedlichen Datenquellen in der Bundesauswertungsstelle (BAS) zur Berechnung der Qualitätsindikatoren sowie für den Versand der Rückmeldeberichte bzw. Jahresberichte von der BAS an die Leistungserbringer (LE) bildet die systemweit einheitliche Pseudonymisierung die Grundlage. Die DeQS-RL fordert die systemweit einheitliche Leistungserbringerpseudonymisierung (LE-Pseudonymisierung) in § 3 Abs. 2 Satz 2 der Anlage zu Teil 1 der Richtlinie:

> Für die Pseudonymisierung nutzen die Datenannahmestellen nach Teil 1 § 9 der Richtlinie das vom Institut nach § 137a SGB V vorgeschlagene Verfahren zur systemweit einheitlichen Leistungserbringerpseudonymisierung, welches sicherstellt, dass die Datenannahmestellen den gleichen leistungserbringeridentifizierenden Daten jeweils das gleiche Pseudonym zuordnen.
>
> — DeQS-RL, § 3 Abs. 2 Satz 2 der Anlage zu Teil 1

Die Erzeugung von kryptografischen Schlüsseln und von Zertifikaten sowie die Pseudonymisierung und Depseudonymisierung von leistungserbringeridentifizierenden Daten werden mithilfe des Leistungserbringerpseudonymisierungsprogramms durchgeführt.

Auf der Website https://iqtig.org/spezifikationen/ergaenzende-downloads/pseudonymisierung kann die aktuelle Version des Programms zusammen mit der Anwenderdokumentation als ZIP-Archiv heruntergeladen werden.

2.1 Verschlüsselungsverfahren zur Pseudonymisierung

Die Erstellung der LE-Pseudonyme wird mittels eines asymmetrischen (Public-Key-) Verschlüsselungsverfahrens durchgeführt. Im Gegensatz zu symmetrischen Verschlüsselungsverfahren wird kein gemeinsamer geheimer Schlüssel benötigt, den beide Kommunikationspartner kennen. Bei der asymmetrischen Verschlüsselung wird ein Schlüsselpaar, bestehend aus einem privaten Schlüssel und einem öffentlichen Schlüssel, generiert. Die Schlüssel sind gemeinsam mit weiteren Informationen in Zertifikate eingebettet.

Der private Schlüssel wird bei der Erzeugung des Schlüsselpaars mit einem Passwort geschützt. Der öffentliche Schlüssel wird zwischen der DAS, die ihn erzeugt hat, und der DAS KK ausgetauscht, um für Routinedaten der Krankenkassen einheitliche LE-Pseudonyme zu erzeugen. Mithilfe des öffentlichen Schlüssels können nun Daten verschlüsselt und sicher an den Besitzer des privaten Schlüssels übermittelt werden. Mit dem privaten Schlüssel und dem Passwort können die empfangenen Daten entschlüsselt werden. Die öffentlichen Schlüssel sind in Zertifikate eingebettet, die zusätzliche Metadaten enthalten.

Für die LE-Pseudonymisierung wird ein deterministisches, asymmetrisches Verschlüsselungsverfahren gebraucht. Verwendung findet der RSA-Verschlüsselungsalgorithmus von Legion of the Bouncy Castle Inc [https://www.bouncycastle.org]. Dieser Algorithmus erfüllt sowohl die Anforderungen an die systemweit einheitliche Pseudonymisierung als auch die vom Bundesamt für Sicherheit in der Informationstechnik (BSI) [https://www.bsi.bund.de] empfohlenen Kriterien. Die deterministische

Verschlüsselung nimmt in diesem Zusammenhang einen besonderen Stellenwert ein, da sichergestellt ist, dass die Verschlüsselung derselben LE-identifizierenden Daten mit dem gleichen öffentlichen Schlüssel bei verschiedenen DAS immer das gleiche Ergebnis produziert.



Die Zertifikate mit den öffentlichen Schlüsseln dürfen nur den im Datenfluss vorgesehenen Stellen zur Verfügung gestellt werden. Es ist z. B. falsch, diese Zertifikate auf einer Website oder mit der Spezifikation zu veröffentlichen. In diesem Fall würde durch die Nutzung der deterministischen Verschlüsselung ein Besitzer von IK/BSNR die jeweiligen Pseudonyme erstellen können. Das Zertifikat, welches den privaten Schlüssel enthält, dient nur dazu, Rückmeldeberichte mit Sozialdaten dem richtigen Leistungserbringer zuordnen zu können, und darf daher Dritten nicht zur Verfügung gestellt werden.

2.2 Anwendungsbereich

Bei dem Pseudonymisierungsprogramm in der Version 3.x handelt es sich um eine Neuentwicklung. Es wird Funktionalität für die Leistungserbringerpseudonymisierung ab dem Jahr 2021 bereitgestellt. Die erste Ausbaustufe umfasst eine Konsolenanwendung für die programmatische Integration der Leistungserbringerpseudonymisierung in nicht Java-basierte Datenflüsse bei den Datenannahmestellen. Die zweite Ausbaustufe in Gestalt von Version 4.x enthält inzwischen auch die Funktionen des PSP-Alt in Version 2.x (veröffentlichte Version 11) (Siehe auch Migrationspfade in Kapitel 5). Folgende Funktionen werden unterstützt:

- Erstellung einer Zuordnung alter zu neuen Pseudonymen bei Veränderungen in der Struktur der Leistungserbringer.
  (S. ***Anwendungsfall Mapping***),
- Erstellung einer Liste aller verwendeter Pseudonyme
  (S. ***Anwendungsfall Positivliste***),
- Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von Klartext-Kennungen
  (S. ***Anwendungsfall Pseudonymisierung von CSV-Datei zu CSV-Datei***).
- Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von IAnr-Kennungen
  (S. ***Anwendungsfall Pseudonymisierung von IAnr-Kennungen (CSV-Datei zu CSV-Datei)***).
- Depseudonymisierung einer Eingabeliste von CSV-Datei zu CSV-Datei
  (S. ***Anwendungsfall Depseudonymisierung von CSV-Datei zu CSV-Datei***).
- Depseudonymisierung einer IAnr-Pseudonyme-Liste von CSV-Datei zu CSV-Datei
  (S. ***Anwendungsfall Depseudonymisierung von IAnr-Pseudonymen von CSV-Datei zu CSV-Datei***).
- Erstellung einer Depseudonymisierung bei Anforderung (Ausgabe JSON)
  (S. ***Anwendungsfall Depseudonymisierung***).
- Erstellung einer Zuordnung alter zu neuen Pseudonymen bei einem Wechsel des Schlüssels
  (S. ***Anwendungsfall Zertifikatswechsel***).
- Erzeugung eines Schlüsselpaares bestehend aus einem Zertifikat und einem privaten Schlüssel
  (S. ***Anwendungsfall Schlüsselpaarerzeugung***).

2.3 Eigenschaften der Leistungserbringerpseudonymisierung

Das Pseudonymisierungsprogramm nutzt einen deterministischen, asymmetrischen RSA-Verschlüsselungsalgorithmus mit einer Schlüssellänge von 2048 Bit zur Erzeugung von Pseudonymen.

2.3.1 Institutskennzeichen-/Betriebsstättennummer-Pseudonym

Ein Institutskennzeichen (IK) / Betriebsstättennummer (BSNR)-Pseudonym ist folgendermaßen aufgebaut:

- Art der Datenannahme (LQS/KV),
- Bundesland/Region,
- spezifizierte Trennzeichen ($$, ##),
- mit dem öffentlichen Schlüssel verschlüsselte IK/BSNR.

Beispiel: `LQS$$HH##fL1PqnoBllAaM2bXSXvf1MOnhp4NhPx2sKqnlng`

2.3.2 Standortpseudonym

Ein Standortpseudonym ist folgendermaßen aufgebaut:

- Art der Datenannahme (muss LQS sein),
- Bundesland/Region,
- spezifizierte Trennzeichen (**, ##),
- mit dem öffentlichen Schlüssel verschlüsselte StandortId.

Beispiel: `LQS**HH##fL1PqnoBllAaM2bXSXvf1MOnhp4NhPx2sKqnlng`

2.3.3 Lebenslange Arztnummer

Ein lebenslange-Arztnummer-Pseudonym ist folgendermaßen aufgebaut:

- Art der Datenannahme KV (muss KV sein),
- Bundesland/Region,
- spezifizierte Trennzeichen (**, ##),
- mit dem öffentlichen Schlüssel verschlüsselte lebenslange Arztnummer.

Beispiel: `KV**HH##fL1PqnoBllAaM2bXSXvf1MOnhp4NhPx2sKqnlng`

2.3.4 Durchführung der Pseudonymisierung

Die einheitliche Pseudonymisierung ist von allen Datenannahmestellen gemäß DeQS-RL durchzuführen. Für die Krankenhäuser ist das Institutskennzeichen (IK) und die StandortId [offizielle StandortId des bundesweiten Verzeichnisses der Standorte der nach § 108 SGB V zugelassenen Krankenhäuser und ihrer Ambulanzen] zu pseudonymisieren, im kollektivvertraglichen Bereich ist stattdessen die Betriebsstättennummer (BSNR) (XML-Element BSNRAMBU) zu pseudonymisieren. Die DAS für die Krankenkassen (DAS-KK) besitzen keine eigenen Zertifikate und müssen deshalb die öffentlichen Zertifikate der DAS auf Landesebene nutzen. Dazu müssen diese DAS entscheiden, ob es sich bei den leistungserbringeridentifizierenden Daten um ein IK und eine StandortId oder BSNR handelt. Die IK und StandortIds werden mit dem Schlüssel der LKG (ehemals LQS) verschlüsselt, die BSNRs mit dem Schlüssel der KV. Zusätzlich muss auch das Bundesland bekannt sein, damit der richtige Schlüssel auf Landesebene ausgewählt werden kann.

2.3.5 Durchführung der Depseudonymisierung

Für die Erstellung von Auswertungen im Rahmen der Richtlinie zu planungsrelevanten Qualitätsindikatoren (plan. QI-RL) [https://www.g-ba.de/richtlinien/91] sowie zur Durchführung der Stellungnahmeverfahren auf Bundesebene mit rechnerisch auffälligen Leistungserbringern im Rahmen der DeQS-RL ist eine Depseudonymisierung der Leistungserbringer gegenüber der Bundesauswertungsstelle erforderlich. Für Krankenhäuser ist das Pseudonym des IK und das Pseudonym der StandortId zu depseudonymisieren.

Für Leistungserbringer aus dem kollektivvertraglichen Bereich KV muss derzeit keine Depseudonymisierung vorgenommen werden, da die Zuständigkeit für die relevanten Qualitätssicherungsverfahren ausschließlich bei den Ländern liegt.

3 Installation

3.1 Systemvoraussetzungen

Software

- JRE von OpenJDK 21 (siehe https://jdk.java.net/java-se-ri/21)

Das Pseudonymisierungsprogramm ist lauffähig ab Java 21 oder neuer.

3.2 Programmbestandteile

Das Pseudonymisierungsprogramm besteht aus einem ausführbaren Java-Programm in Form einer JAR-Datei und dem API-Modul, das in eine Java-Anwendung eingebunden werden kann.

3.3 Installation

Das Pseudonymisierungsprogramm wird als ZIP-Archiv ausgeliefert, welches unter einem beliebigen Pfad entpackt werden kann. Darin enthalten sind:

- die Konsolenanwendung als ausführbares Java-Programm,
- das Java-API-Modul,
- das Anwenderhandbuch im PDF-Format.
- die Datenflussspezifikation im PDF-Format.

Entwickelt und getestet wurde das Pseudonymisierungsprogramm unter Windows 11 in der Programmiersprache Java. Zur Ausführung von Java-Programmen wird das Java Runtime Environment (JRE) benötigt.

Zur Entwicklung wurde OpenJDK 25 verwendet.

3.4 Aktualisierung

Das Pseudonymisierungsprogramm wird in unregelmäßigen Intervallen Updates erhalten. Gründe für Updates können Fehlerbehebungen, Programmverbesserungen sowie die Anpassung an neue oder geänderte QS-Verfahren sein. Eine alte Programmversion kann durch die neue Version ersetzt werden, indem die JAR-Datei ausgetauscht wird. Es besteht auch die Möglichkeit, mehrere Programmversionen parallel zu nutzen.

4 Programmaufbau

Das Pseudonymisierungsprogramm in der Version 4.x wird neben der Möglichkeit, Funktionen über die Java-API auszuführen, ausschließlich als Konsolenanwendung (ohne grafische Benutzeroberfläche) bereitgestellt, deren Funktionsweise in den nächsten Unterkapiteln genauer erläutert wird.

4.1 Konsolenanwendung (CLI)

Der folgende Abschnitt zeigt die Bedienung des PSP-CLI. CLI ist die Abkürzung für Command Line Interface und bezeichnet die Kommandozeilen-Schnittstelle der Anwendung. Über diese Schnittstelle können dem PSP Befehle in Textform z.B. in der Windows-Kommandozeile übergeben werden. Dies kann durch manuelle Eingabe auf der Tastatur oder durch den Aufruf der entsprechenden Befehle in einem Batch-Skript erfolgen. Wie die Aufrufe im Detail strukturiert sein müssen, zeigen die folgenden Anwendungsfälle.

Die Datenannahmestellen können das PSP-CLI in eine „Nicht-Java"-Datenverarbeitungskette integrieren, um Leistungserbringerkennungen zu pseudonymisieren.

Durch Funktionsparameter und Kommandozeilenoptionen wird die Anwendung entsprechend konfiguriert. Folgende Anwendungsfälle werden momentan unterstützt:

Anwendungsfall `Mapping`
Übermittlung der Änderungen von Leistungserbringern an die BAS

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar mapping
             -y 2021
             -i C:\pseudo\input.csv
             -o C:\pseudo
             -c C:\pseudo\key\LQS$$HH20150501.cer
```

Anwendungsfall `Positivliste`
Übermittlung aller Leistungserbringerpseudonyme an die BAS

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar positivliste
             -y 2021
             -i C:\pseudo\input.csv
             -o C:\pseudo
             -c C:\pseudo\key\LQS$$HH20150501.cer
```

Anwendungsfall
Pseudonymisierung - Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von Klartext-Kennungen (von CSV-Datei zu CSV-Datei)

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar pseudonymisierung_csv
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -c C:\pseudo\key\LQS$$HH20150501.cer
```

Anwendungsfall
Pseudonymisierung - Erstellung einer Liste von lAnr-Pseudonymen aus einer Eingabeliste von lAnr-Kennungen (von CSV-Datei zu CSV-Datei)

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar pseudonymisierung_lanr_csv
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -c C:\KV$$HH20150501.cer
```

Anwendungsfall
Depseudonymisierung - Depseudonymisierung einer Eingabeliste im CSV-Format von Pseudonymen mit Ausgabe als CSV-Datei

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar depseudonymisierung_csv
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -k C:\pseudo\key\LQS$$HH20150501.p12
              -p geheimesPasswort
```

Anwendungsfall
Depseudonymisierung - Depseudonymisierung einer Eingabeliste von lAnr-Pseudonymen im CSV-Format mit Ausgabe als CSV-Datei

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar depseudonymisierung_lanr_csv
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -k C:\pseudo\key\KV$$HH20150501.p12
              -p geheimesPasswort
```

Anwendungsfall `Depseudonymisierung`
Übermittlung von depseudonymisierten Leistungserbringerkennungen an die BAS

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar depseudonymisierung
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -k C:\pseudo\key\LQS$$HH20150501.p12
              -p geheimesPasswort
```

Anwendungsfall
Zertifikatswechsel - Übermittlung der Änderungen an die BAS bei einem Zertifikatswechsel

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar zertifikatswechsel
              -y 2021
              -i C:\pseudo\input.csv
              -o C:\pseudo
              -co C:\pseudo\key\LQS$$HH20150501.cer
              -cn C:\pseudo\key\LQS$$HH20210101.cer
```

Anwendungsfall
Schlüsselpaarerzeugung - Erzeugung eines Zertifikates und einem privaten Schlüssel

Beispiel eines Kommandozeilenaufrufes:

```
java -jar psp-cli.jar keys
              -d KV
              -l BB
              -f 31.01.2023
              -t 31.01.2028
              -p A2345678
              -o C:\pseudo
```



Die Erzeugung der Zertifikatsdatei, welche den öffentlichen Schlüssel enthält, kann mit der aktuellen Version des Pseudonymisierungsprogramms vorgenommen werden. Bisher genutzte Zertifikatsdateien, die mit dem Pseudonymisierungsprogramm in Version 2.x erzeugt wurden, können weiterhin verwendet werden.

4.1.1 Mapping - Übermittlung der Änderungen von Leistungserbringern an die BAS

Diese Funktion dient dazu, der BAS Informationen über Veränderungen bei den Leistungserbringern (Leistungserbringermapping) zur Verfügung zu stellen. Die DAS erzeugt zunächst eine CSV-Datei, welche das Leistungserbringermapping im Klartext enthält. Diese wird mithilfe des Pseudonymisierungsprogramms in eine pseudonymisierte JSON-Datei überführt, welche anschließend an die BAS übermittelt werden kann. Die nachfolgend beschriebene Funktion validiert und pseudonymisiert die Eingangs-Datei und legt das Ergebnis in einem vom Anwender definierten Ausgabeverzeichnis ab.

Funktionsparameter: `m` oder `mapping`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen der DAS zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	typ_aenderung
1	modul
2	ikbs_kennung_alt
3	sto_kennung_alt
4	ikbs_kennung_neu
5	sto_kennung_neu
6	geschlossen_ab

Es folgt eine genaue Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen, sowie deren Wertmengen.

Spalte `typ_aenderung`

Die Änderungen von Leistungserbringern können von unterschiedlicher Natur sein. Um diese unterscheiden zu können, ist ein Änderungstyp anzugeben.

- `V` Beschreibt eine Veränderung bei dem Leistungserbringer.
  Beispiel 1: Ein Standort wird einem anderen Krankenhaus zugeordnet.
  Beispiel 2: Die Fachabteilung Herzchirurgie wird von einem Standort zu einem anderen verlegt.
- `S` Kennzeichnet die Schließung einer Betriebsstätte, eines Standortes oder einer Abteilung. Ist dieser Wert angegeben, so muss die Spalte `geschlossen_ab` gefüllt sein. Die Spalten `ikbs_kennung_neu` sowie `sto_kennung_neu` bleiben leer. Bezieht sich der Eintrag auf die Schließung eines Standortes, so wird in der Spalte `modul` * angegeben. Sind nur einzelne Abteilungen von der Schließung betroffen so werden für die betroffenen Module separate Einträge angelegt.

Spalte `modul`

Das Erfassungsmodul, für welches das Mapping angewendet werden soll. Hier ist auch ein * zulässig. Dies bedeutet, dass für alle Erfassungsmodule bei der BAS Mappingeinträge erzeugt werden.

Die Expansion des Sternes erfolgt in Abhängigkeit des Spezifikationsjahres. Dies bedeutet, dass ein Mappingeintrag, der für das Jahr 2021 gemeldet wird, nur auf die Erfassungsmodule expandiert wird, die bereits 2021 im Rahmen der DeQS-Richtlinie ausgewertet wurden.

Für den LKG-Bereich erfolgt eine Expansion ausschließlich auf die für den LKG-Bereich relevanten Erfassungsmodule und für den KV-Bereich erfolgt eine Expansion ausschließlich auf die für den KV-Bereich relevanten Erfassungsmodule.

Mögliche Erfassungsmodule sind:

Kürzel	Beschreibung
09/1	Herzschrittmacher-Implantation
09/2	Herzschrittmacher-Aggregatwechsel
09/3	Herzschrittmacher-Revision/-Systemwechsel/-Explantation
09/4	Implantierbare Defibrillatoren-Implantation
09/5	Implantierbare Defibrillatoren-Aggregatwechsel
09/6	Implantierbare Defibrillatoren-Revision/-Systemwechsel/-Explantation
10/2	Karotis-Revaskularisation
15/1	Gynäkologische Operationen
16/1	Geburtshilfe
17/1	Hüftgelenknahe Femurfraktur mit osteosynthetischer Versorgung
18/1	Mammachirurgie
APSY	Ambulante Psychotherapie
CHE	Cholezystektomie
DEK	Dekubitusprophylaxe
DIAL	Dialyse

Kürzel	Beschreibung
HCH	Koronarchirurgie und Eingriffe an Herzklappen
HEP	Hüftendoprothesenversorgung
HTXM	Herztransplantation, Herzunterstützungssysteme/Kunstherzen
KEP	Knieendoprothesenversorgung
LLS	Leberlebendspende
LTX	Lebertransplantation
LUTX	Lungen- und Herz-Lungentransplantation
NEO	Neonatologie
NLS	Nierenlebendspende
PCI	Perkutane Koronarintervention und Koronarangiographie
PNEU	Ambulant erworbene Pneumonie
PNTX	Nieren- und Pankreas- (Nieren-) transplantation
SEP	Diagnostik und Therapie der Sepsis: Fallbezogene QS-Dokumentation

Spalte `ikbs_kennung_alt`

Die ursprüngliche Betriebsstättennummer oder das ursprüngliche Institutskennzeichen des Leistungserbringers.

Spalte `sto_kennung_alt`

Die ursprüngliche Standortkennung als 6-stellige StandortId aus dem bundesweiten Standortregister beginnend mit 77.

Bei Mappingeinträgen, die sich auf eine Betriebsstätte eines niedergelassenen Arztes beziehen (KV-Sektor), bleibt dieses Feld leer.

Spalte `ikbs_kennung_neu`

Die neue Betriebsstättennummer oder das neue Institutskennzeichen des Leistungserbringers.

Dieses Feld bleibt leer, wenn über den Mappingeintrag die Schließung (Typ `S`) eines Leistungserbringers mitgeteilt wird.

Spalte `sto_kennung_neu`

Die neue 6-stellige StandortId aus dem bundesweiten Standortregister beginnend mit 77.

Bei Mappingeinträgen, die sich auf eine Betriebsstätte eines niedergelassenen Arztes beziehen (KV-Sektor), bleibt dieses Feld leer.

Dieses Feld bleibt ebenfalls leer, wenn über den Mappingeintrag die Schließung (Typ `S`) eines Leistungserbringers mitgeteilt wird.

Spalte `geschlossen_ab`

Wird über den Mappingeintrag die Schließung eines Leistungserbringers mitgeteilt, wird in dieses Feld das Datum des ersten Tages, an dem der Leistungserbringer geschlossen ist, eingetragen. Das Datum ist im Format dd.MM.yyyy anzugeben.

Erfolgt hier ein Eintrag, so müssen die Felder `ikbs_kennung_neu` sowie `sto_kennung_neu` leer bleiben.

Ergänzung der Eingabedatei und Übertragung der Ausgabedatei

Die DAS kann die Eingabedatei während eines Jahres immer wieder ergänzen und die daraus erzeugte Ausgabedatei erneut an die BAS übertragen. Es können aber auch mehrere Eingabedateien für ein Jahr verwendet werden und somit auch mehrere Ausgabedateien übertragen werden. Wichtig ist, dass sich die für ein Jahr gemeldeten Mappingeinträge nicht widersprechen. Eine alte Leistungserbringerkennung kann nicht innerhalb eines Jahres auf verschiedene neue Leistungserbringerkennungen gemappt werden. Innerhalb jeder Eingabedatei wird diese auf Konsistenz geprüft und die Weiterverarbeitung ggf. mit einer entsprechenden Fehlermeldung abgebrochen. Überträgt die DAS mehrere Dateien pro Jahr, in denen jeweils nur ein Ausschnitt enthalten ist, so können die Konsistenzprüfungen erst bei der weiteren Verarbeitung bei der BAS erfolgen. Um einen möglichst reibungslosen Ablauf bei DAS und BAS zu ermöglichen, wird empfohlen eine Eingabedatei pro Jahr fortlaufend zu ergänzen.

Das Spezifikationsjahr

Der Anwender gibt das Jahr an, in dem die Veränderung bei dem Leistungserbringer stattgefunden hat.

Kommandozeilenoption: `-y` oder `--year`

Maßgeblich dafür, ob ein bestimmter QS-Datensatz von einem Mappingeintrag erfasst wird, ist das im QS-Datensatz angegebene Spezifikationsjahr.

Die QS-Datensätze des betreffenden Leistungserbringers, die in dem angegebenen Spezifikationsjahr und früheren Spezifikationsjahren mit der alten Kennung geliefert wurden, werden unter der neuen Kennung zusammengeführt. Der kleinste zulässige Wert ist 2020.

Über dieses Vorgehen werden Überlieger miterfasst, auch wenn die QS-Datensätze erst im Folgejahr in die Auswertung einfließen.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem das pseudonymisierte Leistungserbringermapping abgelegt wird.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem wird ein ZIP-Archiv abgelegt, welches eine JSON-Datei mit dem pseudonymisierten Leistungserbringermapping enthält.

Diese Datei darf *nachträglich nicht verändert* werden.

Das entstandene ZIP-Archiv lädt die DAS über den Teilnehmerdialog der BAS zur weiteren Verarbeitung hoch.

Parallel zum ZIP-Archiv wird eine Quittungsdatei im CSV-Format erzeugt. Die Quittungsdatei enthält

eine Kopie der Eingangsdaten, welche zusätzlich um die vorangestellte Zeilennummer ergänzt wurde. Diese Datei darf *nicht an die BAS übertragen* werden und sollte Dritten - wie auch die Eingangsdatei selbst - nicht zugänglich gemacht werden.

Die Zertifikatsdatei

Die Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die Kennungen der Leistungserbringer zu pseudonymisieren.

Kommandozeilenoption: `-c` oder `--cert`

Über diese Option wird der Pfad zur Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Zum Zeitpunkt der Verarbeitung muss die verwendete Zertifikatsdatei gültig sein (aktuelles Datum zwischen Start- und Enddatum des Zertifikates).

4.1.2 Positivliste - Übermittlung aller Leistungserbringerpseudonyme an die BAS

Diese Funktion dient dazu der BAS sämtliche bei der DAS bekannten Leistungserbringerkennungen in pseudonymisierter Form zu übermitteln. Die DAS erzeugt zunächst eine CSV-Datei, welche die Leistungserbringerkennungen im Klartext enthält. Diese wird mithilfe des Pseudonymisierungsprogramms in eine pseudonymisierte JSON-Datei überführt, welche anschließend an die BAS übermittelt werden kann. Die nachfolgend beschriebene Funktion validiert und pseudonymisiert die Eingangs-Datei und legt das Ergebnis in einem vom Anwender definierten Ausgabeverzeichnis ab.

Funktionsparameter: `p` oder `positivliste`

Das Spezifikationsjahr

Das Spezifikationsjahr, für welches die Leistungserbringerkennungen gültig sind.

Kommandozeilenoption: `-y` oder `--year`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen der DAS zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	verfahren
1	ikbs
2	stoid

Es folgt eine genaue Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen, sowie deren Wertmengen.

Spalte `verfahren`

Das Pseudonymisierungsverfahren, für welches das Pseudonym erzeugt werden soll. Hier ist auch ein * zulässig. Dies bedeutet, dass für alle Verfahren Pseudonyme erzeugt werden.

Die Expansion des Sternes erfolgt in Abhängigkeit des Spezifikationsjahres. Dies bedeutet, dass für eine für das Jahr 2021 gemeldete Leistungserbringerkennung nur für die Pseudonymisierungsverfahren Pseudonyme erzeugt werden, die bereits 2021 im Rahmen der DeQS-Richtlinie ausgewertet wurden.

Spalte `ikbs`

Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers.

Spalte `stoid`

Die StandortId, welche eine 6-stellige Zahl, beginnend mit 77, aus dem bundesweiten Standortregister sein muss.

Bei Einträgen, die sich auf eine Betriebsstätte eines niedergelassenen Arztes beziehen (KV-Sektor), bleibt dieses Feld leer.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem die Positivliste abgelegt wird.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem wird ein ZIP-Archiv abgelegt, welches eine JSON-Datei mit den pseudonymisierten Leistungserbringerkennungen enthält.

Diese Datei darf *nachträglich nicht verändert* werden.

Das entstandene ZIP-Archiv übermittelt die DAS zur weiteren Verarbeitung an die BAS.

Parallel zum ZIP-Archiv wird eine Quittungsdatei im CSV-Format erzeugt. Die Quittungsdatei enthält eine Kopie der Eingangsdaten, welche zusätzlich um die vorangestellte Zeilennummer ergänzt wurde. Diese Datei darf *nicht an die BAS übermittelt* werden und sollte Dritten - wie auch die Eingangsdatei selbst - nicht zugänglich gemacht werden.

Die Zertifikatsdatei

Die Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die Kennungen der Leistungserbringer zu pseudonymisieren.

Kommandozeilenoption: `-c` oder `--cert`

Über diese Option wird der Pfad zur Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Zum Zeitpunkt der Verarbeitung muss die verwendete Zertifikatsdatei gültig sein (aktuelles Datum zwischen Start- und Enddatum des Zertifikates).

4.1.3 Pseudonymisierung - Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von Klartext-Kennungen

Diese Funktion dient dazu, aus einer in Form einer CSV-Datei übergebenen Liste von IkBs- bzw. Stold-Kennungen eine Liste von entsprechenden IkBs- bzw. Stold-Pseudonymen zu generieren. Jede Zeile der übergebenen CSV-Datei (die Kopfzeile ausgenommen) besteht aus einem Datensatz, der das Pseudonymisierungsverfahren, eine IkBs-Kennung und/oder eine Stold-Kennung (mindestens ein Feld muss gefüllt sein) enthält. Pro Zeile wird in der resultierenden Ausgabe-CSV-Datei eine Ausgabezeile geschrieben, die die Eingabefelder enthält plus des generierten IkBs-Pseudonyms und des generierten Stold-Pseudonyms. Pseudonymisiert wird mit einem als Kommandozeilenoption übergebenen Zertifikat.

Funktionsparameter: `pc` oder `pseudonymisierung_csv`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein. Die Zeilen der Eingabedatei dürfen sich auf verschiedene Pseudonymisierungsverfahren beziehen.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	pseudonymisierungsverfahren
1	ikbs_kennung
2	stoid_kennung

Es folgt eine Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen.

Spalte `pseudonymisierungsverfahren`

Das Pseudonymisierungsverfahren, für welches das Pseudonym erzeugt werden soll. (Eine Expansion mit * ist nicht zulässig.)

Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) angegeben werden.

Mögliche Pseudonymisierungsverfahren sind:

Kürzel	Richtlinie	Beschreibung
DK	oKFE	Darmkrebs
ZK	oKFE	Zervixkarzinom
AMBPT	DeQS	Ambulante Psychotherapie
CHE	DeQS	Cholezystektomie
DEK	DeQS	Dekubitusprophylaxe
GYN	DeQS	Gynäkologische Operationen
HCH	DeQS	Koronarchirurgie und Eingriffe an Herzklappen
HGV	DeQS	Hüftgelenkversorgung
HSMDEF	DeQS	Versorgung mit Herzschrittmachern und implantierbaren Defibrillatoren
HTXM	DeQS	Herztransplantationen und Implantation von Herzunterstützungssystemen/Kunstherzen
KAROT	DeQS	Karotis-Rekonstruktion
KEP	DeQS	Knieendoprothesen
LLTX	DeQS	Lebertransplantation und Leberlebendspende
LUTX	DeQS	Lungentransplantationen und Herz-Lungen-Transplantationen
MAMMA	DeQS	Mammachirurgie
NET	DeQS	Nierenersatztherapie bei chronischem Nierenversagen einschließlich Pankreastransplantationen
NLS	DeQS	Nierenlebendspenden
PCI	DeQS	Perkutane Koronarintervention und Koronarangiographie
PERI	DeQS	Perinatalmedizin
PNEU	DeQS	Ambulant erworbene Pneumonie
SEP	DeQS	Sepsis

Spalte `ikbs_kennung`

Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers.

Spalte `stoid_kennung`

Die StandortId, welche eine 6-stellige Zahl, beginnend mit 77, aus dem bundesweiten Standortregister sein muss.

Bei Einträgen, die sich auf eine Betriebsstätte eines niedergelassenen Arztes beziehen (KV-Sektor), bleibt dieses Feld leer.

Mindestens eines der Felder "ikbs_kennung" oder "stoid_kennung" muss in jeder Zeile gefüllt sein.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem die pseudonymisierten Leistungserbringerkennungen als CSV-Datei abgelegt werden.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem Verzeichnis wird die resultierende CSV-Datei ausgegeben.

Die Zertifikatsdatei

Die Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die Kennungen der Leistungserbringer zu pseudonymisieren.

Kommandozeilenoption: `-c` oder `--cert`

Über diese Option wird der Pfad zur Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Zum Zeitpunkt der Verarbeitung muss die verwendete Zertifikatsdatei gültig sein (aktuelles Datum zwischen Start- und Enddatum des Zertifikates).

Die ausgegebene CSV-Datei

Im Ausgabeverzeichnis wird als Ergebnis des Programmlaufes eine Datei erzeugt, die folgendem Benennungsschema folgt:

```
<Timestamp_yyyyMMddhhmmss>_pseudonymisierung_csv_result_<inputFile>.csv
```

Hierbei steht <inputFile> für den Namen der Eingabedatei ohne Erweiterung.

Die Ausgabedatei enthält mindestens folgende Spalten

Index	Spaltenname
0	pseudonymisierungsverfahren
1	ikbs_kennung
2	stoid_kennung
3	ikbs_pseudonym
4	stoid_pseudonym

Die Spalten 0 - 2 entsprechen der Input-Zeile in der Eingabe-CSV-Datei. Dazugekommen sind die Spalten "ikbs_pseudonym" und "stoid_pseudonym", die jeweils das Pseudonymisierungsergebnis der Spalten 1 bzw. 2 enthalten. Die ausgegebene CSV-Datei kann nach den genannten Spalten weitere Spalten enthalten. Nämlich dann, wenn die Eingabedatei nach den Spalten 0 - 2 noch weitere Spalten enthalten hat. Diese werden ohne weitere Verarbeitung in die Ausgabe-CSV-Datei nach der Spalte mit Index 4 "stoid_pseudonym" angefügt.

4.1.4 Pseudonymisierung - Erstellung einer Liste von lAnr-Pseudonymen aus einer Eingabeliste von lAnr-Kennungen

Diese Funktion dient dazu, aus einer in Form einer CSV-Datei übergebenen Liste von lAnr-Kennungen eine Liste von entsprechenden lAnr-Pseudonymen zu generieren. Jede Zeile der übergebenen CSV-Datei (die Kopfzeile ausgenommen) besteht aus einem Datensatz, der das Pseudonymisierungsverfahren und eine lAnr-Kennung enthält. Pro Zeile wird in der resultierenden Ausgabe-CSV-Datei eine Ausgabezeile geschrieben, die die Eingabefelder enthält plus des generierten lAnr-Pseudonyms. Pseudonymisiert wird mit einem als Kommandozeilenoption übergebenen Zertifikat. Dabei wird die Operation nur durchgeführt, wenn die Datenannahmestelle des Zertifikates "KV" ist, ansonsten bricht die Verarbeitung mit einem entsprechenden Fehler ab.

Funktionsparameter: `plc` oder `pseudonymisierung_lanr_csv`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein. Die Zeilen der Eingabedatei dürfen sich auf verschiedene Pseudonymisierungsverfahren beziehen.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	pseudonymisierungsverfahren
1	lanr_kennung

Es folgt eine Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen.

Spalte `pseudonymisierungsverfahren`

Das Pseudonymisierungsverfahren, für welches das Pseudonym erzeugt werden soll. (Eine Expansion mit * ist nicht zulässig.) Dieses Feld muss in jeder Zeile gefüllt sein.

Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) angegeben werden.

Spalte `lanr_kennung`

Die lebenslange Arztnummer, die pseudonymisiert werden soll. Dieses Feld muss in jeder Zeile gefüllt sein.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem die pseudonymisierten lAnr-Kennungen als CSV-Datei abgelegt werden.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem Verzeichnis wird die resultierende CSV-Datei ausgegeben.

Die Zertifikatsdatei

Die Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die lebenslangen Arztnummern zu pseudonymisieren.

Kommandozeilenoption: `-c` oder `--cert`

Über diese Option wird der Pfad zur Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Zum Zeitpunkt der Verarbeitung muss die verwendete Zertifikatsdatei gültig sein (aktuelles Datum zwischen Start- und Enddatum des Zertifikates).

Die ausgegebene CSV-Datei

Im Ausgabeverzeichnis wird als Ergebnis des Programmlaufes eine Datei erzeugt, die folgendem Benennungsschema folgt:

```
<Timestamp_yyyyMMddhhmmss>_pseudonymisierung_lanr_csv_result_<inputFile>.csv
```

Hierbei steht <inputFile> für den Namen der Eingabedatei ohne Erweiterung.

Die Ausgabedatei enthält mindestens folgende Spalten

Index	Spaltenname
0	pseudonymisierungsverfahren
1	lanr_kennung
2	lanr_pseudonym

Die Spalten 0 - 1 entsprechen der Input-Zeile in der Eingabe-CSV-Datei. Dazugekommen ist die Spalte "lanr_pseudonym", die das Pseudonymisierungsergebnis der Spalte 1 enthält. Die ausgegebene CSV-Datei kann nach den genannten Spalten weitere Spalten enthalten. Nämlich dann, wenn die Eingabedatei nach den Spalten 0 - 1 noch weitere Spalten enthalten hat. Diese werden ohne weitere Verarbeitung in die Ausgabe-CSV-Datei nach der Spalte mit dem Index 2 "lanr_pseudonym" angefügt.

4.1.5 Depseudonymisierung - Depseudonymisierung einer Eingabeliste im CSV-Format von Pseudonymen mit Ausgabe als CSV-Datei

Diese Funktion dient dazu, aus einer in Form einer CSV-Datei übergebenen Liste von IkBs- bzw. Stold-Pseudonymen eine Liste von entsprechenden depseudonymisierten IkBs- bzw. Stold-Kennungen zu generieren. Jede Zeile der übergebenen CSV-Datei (die Kopfzeile ausgenommen) besteht aus einem Datensatz, der ein IkBs-Pseudonym und/oder ein Stold-Pseudonym (mindestens ein Feld muss gefüllt sein) enthält. Pro Zeile wird in der resultierenden Ausgabe-CSV-Datei eine Ausgabezeile geschrieben, die die Eingabefelder enthält plus der erzeugten IkBs-Kennung (Klartext) und der generierten Stold-Kennung (Klartext). Depseudonymisiert wird mit dem privaten Schlüssel, der sich in dem Keystore befindet, dessen Pfad als Kommandozeilenparameter übergeben wird.

Funktionsparameter: `dc` oder `depseudonymisierung_csv`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	ikbs_pseudonym
1	stoid_pseudonym

Es folgt eine Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen.

Spalte `ikbs_pseudonym`

Das mit dem öffentlichen Schlüssel erzeugte Pseudonym des Institutskennzeichens des jeweiligen Leistungserbringers. Während der Depseudonymisierung wird das Pseudonymisierungsverfahren ermittelt. Dieses sollte dasselbe sein wie für das Pseudonym in der Spalte stoid_pseudonym. Falls das nicht zutrifft, erfolgt eine Warnung.

Falls die depseudonymisierte IK- oder BS-Nummer nicht dem erwarteten Format entspricht (9-stelliger Bezeichner, nur aus Ziffern bestehend), wird eine Warnung ausgegeben. Der Prozess wird jedoch nicht abgebrochen. Denn es muss erwartet werden, dass das Eingangs-Pseudonym zuvor mithilfe einer validen IK- oder BS-Nummer erzeugt wurde.

Eine Beschreibung des genauen Aufbaues des IkBs-Pseudonyms findet sich unter 2.3.1.

Spalte `stoid_pseudonym`

Das mit dem öffentlichen Schlüssel erzeugte Pseudonym der StandortId des jeweiligen Leistungserbringers. Während der Depseudonymisierung wird das Pseudonymisierungsverfahren ermittelt. Dieses sollte dasselbe sein wie für das Pseudonym in der Spalte ikbs_pseudonym, ansonsten erfolgt eine Warnung.

Falls die depseudonymisierte StandortId nicht dem erwarteten Format eines Krankenhaus-Standortes entspricht (77xxxx), wird eine Warnung ausgegeben, jedoch wird der Prozess nicht abgebrochen, da erwartet werden muss, dass das Eingangs-Pseudonym zuvor mithilfe einer validen StandortId erzeugt wurde.

Eine Beschreibung des genauen Aufbaues des Stold-Pseudonyms findet sich unter 2.3.2.

Mindestens eines der Felder "ikbs_pseudonym" oder "stoid_pseudonym" muss in jeder Zeile gefüllt sein.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem die depseudonymisierten Leistungserbringerpseudonyme als CSV-Datei abgelegt werden.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem Verzeichnis wird die resultierende CSV-Datei ausgegeben.

Die Schlüsseldatei

Die Schlüsseldatei (Key Store) mit dem *privaten Schlüssel* wird benötigt, um die mit dem öffentlichen Schlüssel erstellten Pseudonyme zu depseudonymisieren.

Kommandozeilenoption: `-k` oder `--key`

Über diese Option wird der Pfad zur Schlüsseldatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Die Gültigkeit des Schlüssels (Start- und Enddatum) wird bei der Depseudonymisierung nicht geprüft.

Das Passwort

Die Schlüsseldatei ist mit einem Passwort gesichert. Für die Depseudonymisierung wird das zur Schlüsseldatei passende Passwort benötigt.

Kommandozeilenoption: `-p` oder `--password`

Über diese Option wird das Passwort zur Schlüsseldatei angegeben.

Die ausgegebene CSV-Datei

Im Ausgabeverzeichnis wird als Ergebnis des Programmlaufes eine Datei erzeugt, die folgendem Benennungsschema folgt:

```
<Timestamp_yyyyMMddhhmmss>_depseudonymisierung_csv_result_<inputFile>.csv
```

Hierbei steht <inputFile> für den Namen der Eingabedatei ohne Erweiterung.

Die Ausgabedatei enthält mindestens folgende Spalten

Index	Spaltenname
0	ikbs_pseudonym
1	stoid_pseudonym
2	pseudonymisierungsverfahren
3	ikbs_kennung
4	stoid_kennung

Die Spalten 0 - 1 entsprechen der Input-Zeile in der Eingabe-CSV-Datei. Dazugekommen sind die Spalten "pseudonymisierungsverfahren", "ikbs_kennung" und "stoid_kennung". Die Spalten 3 bzw. 4 enthalten das Depseudonymisierungsergebnis der Spalten 0 bzw. 1. Das Pseudonymisierungsverfahren wird bei der Depseudonymisierung aus dem IkBs-Pseudonym bzw. dem StoId-Pseudonym extrahiert. Sollten diese verschieden sein, wird das IkBs-Pseudonymisierungsverfahren als Wert für die Spalte "pseudonymisierungsverfahren" verwendet. Die ausgegebene CSV-Datei kann nach den genannten Spalten weitere Spalten enthalten. Nämlich dann, wenn die Eingabedatei nach den Spalten 0 - 1 noch weitere Spalten enthalten hat. Diese werden ohne weitere Verarbeitung in die Ausgabe-CSV-Datei nach der Spalte mit dem Index 4 "stoid_kennung" angefügt.

4.1.6 Depseudonymisierung - Depseudonymisierung einer Eingabeliste von lAnr-Pseudonymen im CSV-Format mit Ausgabe als CSV-Datei

Diese Funktion dient dazu, zu einer in Form einer CSV-Datei übergebenen Liste von lAnr-Pseudonymen eine Liste von entsprechenden depseudonymisierten lAnr-Kennungen zu generieren. Jede Zeile der übergebenen CSV-Datei (die Kopfzeile ausgenommen) besteht aus einem Datensatz, der ein lAnr-Pseudonym enthält. Pro Zeile wird in der resultierenden Ausgabe-CSV-Datei eine Ausgabezeile geschrieben, die das Eingabefeld enthält plus der erzeugten lAnr-Kennung (Klartext). Depseudonymisiert wird mit dem privaten Schlüssel, der sich in dem Keystore befindet, dessen Pfad als Kommandozeilenparameter übergeben wird.

Funktionsparameter: `dlc` oder `depseudonymisierung_lanr_csv`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei enthält in diesem Anwendungsfall genau eine Spalte. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	lanr_pseudonym

Es folgt eine Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen.

Spalte `lanr_pseudonym`

Das mit dem öffentlichen Schlüssel erzeugte Pseudonym der lebenslangen Arztnummer (lAnr). Während der Depseudonymisierung wird das Pseudonymisierungsverfahren ermittelt.

Falls die depseudonymisierte lebenslange Arztnummer nicht dem erwarteten Format entspricht (9-stelliger Bezeichner, nur aus Ziffern bestehend), wird eine Warnung ausgegeben, jedoch wird der Prozess nicht abgebrochen, da erwartet werden muss, dass das Eingangs-Pseudonym zuvor mithilfe einer validen lebenslangen Arztnummer erzeugt wurde.

Eine Beschreibung des genauen Aufbaues des lAnr-Pseudonyms findet sich unter 2.3.3.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem die depseudonymisierten lebenslangen Arztnummern als CSV-Datei abgelegt werden.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem Verzeichnis wird die resultierende CSV-Datei ausgegeben.

Die Schlüsseldatei

Die Schlüsseldatei (Key Store) mit dem *privaten Schlüssel* wird benötigt, um die mit dem öffentlichen Schlüssel erstellten Pseudonyme zu depseudonymisieren.

Kommandozeilenoption: `-k` oder `--key`

Über diese Option wird der Pfad zur Schlüsseldatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Die Gültigkeit des Schlüssels (Start- und Enddatum) wird bei der Depseudonymisierung nicht geprüft.

Das Passwort

Die Schlüsseldatei ist mit einem Passwort gesichert. Für die Depseudonymisierung wird das zur Schlüsseldatei passende Passwort benötigt.

Kommandozeilenoption: `-p` oder `--password`

Über diese Option wird das Passwort zur Schlüsseldatei angegeben.

Die ausgegebene CSV-Datei

Im Ausgabeverzeichnis wird als Ergebnis des Programmlaufes eine Datei erzeugt, die folgendem Benennungsschema folgt:

```
<Timestamp_yyyyMMddhhmmss>_depseudonymisierung_lanr_csv_result_<inputFile>.csv
```

Hierbei steht <inputFile> für den Namen der Eingabedatei ohne Erweiterung.

Die Ausgabedatei enthält mindestens folgende Spalten

Index	Spaltenname
0	lanr_pseudonym
1	pseudonymisierungsverfahren
2	lanr_kennung

Die Spalte 0 entspricht der Input-Spalte in der Eingabe-CSV-Datei. Dazugekommen sind die Spalten "pseudonymisierungsverfahren" und "lanr_kennung". Die Spalte 2 enthält das Depseudonymisierungsergebnis der Spalte 0. Das Pseudonymisierungsverfahren wird bei der Depseudonymisierung aus dem lAnr-Pseudonym extrahiert. Die ausgegebene CSV-Datei kann nach den genannten Spalten weitere Spalten enthalten. Nämlich dann, wenn die Eingabedatei nach der Spalte 0 noch weitere Spalten enthalten hat. Diese werden ohne weitere Verarbeitung in die Ausgabe-CSV-Datei nach der Spalte mit dem Index 2 "lanr_kennung" angefügt.

4.1.7 Depseudonymisierung - Übermittlung von depseudonymisierten Leistungserbringerkennungen an die BAS

Diese Funktion dient dazu ausgewählte Leistungserbringer gegenüber der BAS zu depseudonymisieren. Die DAS erhält hierzu von der BAS eine CSV-Datei, welche die Pseudonyme der zu depseudonymisierenden Leistungserbringer enthält. Diese wird mithilfe des Pseudonymisierungsprogramms in eine JSON-Datei überführt, welche die Leistungserbringerkennungen in pseudonymisierter und depseudonymisierter Form enthält, und welche anschließend an die BAS übermittelt werden kann. Die nachfolgend beschriebene Funktion validiert und depseudonymisiert die Eingangs-Datei und legt das Ergebnis in einem vom Anwender definierten Ausgabeverzeichnis ab.

Für alle anderen Depseudonymisierungen, die nicht zu diesem Anwendungsfall passen, steht die Funktion 4.1.5 Depseudonymisierung - Depseudonymisierung einer Eingabeliste im CSV-Format von Pseudonymen mit Ausgabe als CSV-Datei zu Verfügung.

Funktionsparameter: `d` oder `depseudonymisierung`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile und muss zwingend enthalten sein. Die Zeilen der Eingabedatei dürfen sich auf verschiedene Pseudonymisierungsverfahren beziehen.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen der DAS zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	ikbs_pseudonym
1	stoid_pseudonym

Es folgt eine genaue Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen. Jede Zeile muss in beiden Spalten Pseudonyme enthalten. Leere Felder führen zu einem Fehler.

Spalte `ikbs_pseudonym`

Das mit dem öffentlichen Schlüssel erzeugte Pseudonym des Institutskennzeichens des jeweiligen Leistungserbringers. Während der Depseudonymisierung wird das Pseudonymisierungsverfahren ermittelt. Dieses muss dasselbe sein wie für das Pseudonym in der Spalte stoid_pseudonym.

Falls die depseudonymisierte IK- oder BS-Nummer nicht dem erwarteten Format entspricht (9-stelliger Bezeichner, nur aus Ziffern bestehend), wird eine Warnung ausgegeben, jedoch wird der Prozess nicht abgebrochen, da erwartet werden muss, dass das Eingangs-Pseudonym zuvor mithilfe einer validen IK- oder BS-Nummer erzeugt wurde.

Eine Beschreibung des genauen Aufbaues des IkBs-Pseudonyms findet sich unter 2.3.1.

Spalte `stoid_pseudonym`

Das mit dem öffentlichen Schlüssel erzeugte Pseudonym der StandortId des jeweiligen Leistungserbringers. Während der Depseudonymisierung wird das Pseudonymisierungsverfahren ermittelt. Dieses muss dasselbe sein wie für das Pseudonym in der Spalte ikbs_pseudonym.

Falls die depseudonymisierte StandortId nicht dem erwarteten Format eines Krankenhaus-Standortes entspricht (77xxxx), wird eine Warnung ausgegeben, jedoch wird der Prozess nicht abgebrochen, da erwartet werden muss, dass das Eingangs-Pseudonym zuvor mithilfe einer validen StandortId erzeugt wurde.

Eine Beschreibung des genauen Aufbaues des StoId-Pseudonyms findet sich unter 2.3.2.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem das pseudonymisierte Leistungserbringermapping abgelegt wird.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem wird ein ZIP-Archiv abgelegt, welches eine JSON-Datei mit den depseudonymisierten Leistungserbringerkennungen enthält.

Diese Datei darf *nachträglich nicht verändert* werden.

Das entstandene ZIP-Archiv lädt die DAS über den Teilnehmerdialog der BAS zur weiteren Verarbeitung hoch.

Parallel zum ZIP-Archiv wird eine Quittungsdatei im CSV-Format erzeugt. Die Quittungsdatei enthält eine Kopie der Eingangsdaten, welche zusätzlich um die vorangestellte Zeilennummer ergänzt wurde. Diese Datei darf *nicht an die BAS übertragen* werden und sollte Dritten - wie auch die Eingangsdatei selbst - nicht zugänglich gemacht werden.

Die Schlüsseldatei

Die Schlüsseldatei (Key Store) mit dem *privaten Schlüssel* wird benötigt, um die mit dem öffentlichen Schlüssel erstellten Pseudonyme zu depseudonymisieren.

Kommandozeilenoption: `-k` oder `--key`

Über diese Option wird der Pfad zur Schlüsseldatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Die Gültigkeit des Schlüssels (Start- und Enddatum) wird bei der Depseudonymisierung nicht geprüft.

Das Passwort

Die Schlüsseldatei ist mit einem Passwort gesichert. Für die Depseudonymisierung wird das zur Schlüsseldatei passende Passwort benötigt.

Kommandozeilenoption: `-p` oder `--password`

Über diese Option wird das Passwort zur Schlüsseldatei angegeben.

4.1.8 Zertifikatswechsel

Diese Funktion dient dazu, der BAS im Fall eines Zertifikatswechsels (Wechsel des privaten und öffentlichen Schlüssels) die Zuordnung der alten Pseudonyme zu den neuen Pseudonymen mitzuteilen.

Ein Zertifikatswechsel kann zum Beispiel notwendig bzw. gewünscht sein, wenn:

- die Schlüssel abgelaufen sind (Ablaufdatum überschritten) oder
- Aufgrund eines Sicherheitsvorfalles die Schlüssel getauscht werden sollen.

Für den Zertifikatswechsel werden der alte und der neue öffentliche Schlüssel (.cer) benötigt. Ein neues Schlüsselpaar kann mit dem PSP-Tool wie in Kapitel 4.1.9 beschrieben erzeugt werden.

Achtung: Im LKG-Bereich sind Zertifikatswechsel aufgrund des Wegfalls der dezentralen Pseudonymisierung nicht mehr möglich.

Die DAS erzeugt zunächst eine CSV-Datei, welche das Leistungserbringermapping im Klartext enthält. Diese wird mithilfe des Pseudonymisierungsprogramms in eine pseudonymisierte JSON-Datei überführt, welche anschließend an die BAS übermittelt werden kann. Die nachfolgend beschriebene Funktion validiert und pseudonymisiert die Eingangs-Datei und legt das Ergebnis in einem vom Anwender definierten Ausgabeverzeichnis ab.

Funktionsparameter: `z` oder `zertifikatswechsel`

Das Spezifikationsjahr

Das Spezifikationsjahr, in dem der Zertifikatswechsel durchgeführt wird.

Kommandozeilenoption: `-y` oder `--year`

Die Eingabedatei

Die Eingabedatei wird als semikolonseparierte CSV-Datei erwartet und muss in UTF-8 (ohne BOM) kodiert sein. Die erste Zeile enthält die Kopfzeile. Diese muss zwingend enthalten sein.

Kommandozeilenoption: `-i` oder `--input`

Über diese Option legt der Anwender fest, wo auf seinem System die Eingabedatei abgelegt ist. Der Wert kann ein absoluter oder ein relativer Pfad sein.

Die Eingabedatei muss definierte Spalten (s.u.) in einer vorgegebenen Reihenfolge enthalten. Darüber hinaus können dahinter weitere Spalten enthalten sein, um beispielweise interne Informationen der DAS zu den einzelnen Einträgen zu hinterlegen. Diese werden weder eingelesen noch verarbeitet.

Folgende Spalten sind zu verwenden:

Index	Spaltenname
0	modul
1	ikbs
2	stoid

Es folgt eine genaue Beschreibung der Spalten, die in der Eingabedatei vorhanden sein müssen, sowie deren Wertmengen.

Spalte `modul`

Das Erfassungsmodul, für welches der Zertifikatswechsel angewendet werden soll. In der Regel wird hier bei einem Zertifikatswechsel ein * angegeben. Damit werden für alle Erfassungsmodule bei der BAS Mappingeinträge erzeugt.

Die Expansion des Sternes erfolgt in Abhängigkeit des Spezifikationsjahres. Dies bedeutet, dass ein Mappingeintrag, der für das Jahr 2021 gemeldet wird, nur auf die Erfassungsmodule expandiert wird, die bereits 2021 im Rahmen der DeQS-Richtlinie ausgewertet wurden.

Für den LKG-Bereich erfolgt eine Expansion ausschließlich auf die für den LKG-Bereich relevanten Erfassungsmodule und für den KV-Bereich erfolgt eine Expansion ausschließlich auf die für den KV-Bereich relevanten Erfassungsmodule.

Spalte `ikbs`

Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers.

Spalte `stoid`

Die StandortId, welche eine 6-stellige Zahl, beginnend mit 77, aus dem bundesweiten Standortregister sein muss.

Bei Einträgen, die sich auf eine Betriebsstätte eines niedergelassenen Arztes beziehen (KV-Sektor), bleibt dieses Feld leer.

Das Ausgabeverzeichnis

Der Anwender definiert ein Verzeichnis, in dem das ZIP-Archiv mit dem Ergebnis des Zertifikatwechsels abgelegt wird.

Kommandozeilenoption: `-o` oder `--outdir`

Über diese Option wird der Pfad zu einem Verzeichnis angegeben. In diesem wird ein ZIP-Archiv abgelegt, welches eine JSON-Datei mit den pseudonymisierten Leistungserbringerkennungen enthält.

Diese Datei darf *nachträglich nicht verändert* werden.

Das entstandene ZIP-Archiv übermittelt die DAS zur weiteren Verarbeitung an die BAS.

Parallel zum ZIP-Archiv wird eine Quittungsdatei im CSV-Format erzeugt. Die Quittungsdatei enthält eine Kopie der Eingangsdaten, welche zusätzlich um die vorangestellte Zeilennummer ergänzt wurde. Diese Datei darf *nicht an die BAS übermittelt* werden und sollte Dritten - wie auch die Eingangsdatei selbst - nicht zugänglich gemacht werden.

Die alte Zertifikatsdatei

Die alte Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die Kennungen der Leistungserbringer zu pseudonymisieren.

Kommandozeilenoption: `-co` oder `--certold`

Über diese Option wird der Pfad zur alten Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Die Gültigkeit der alten Zertifikatsdatei (Start- und Enddatum) wird beim Zertifikatswechsel nicht geprüft.

Die neue Zertifikatsdatei

Die neue Zertifikatsdatei mit dem *öffentlichen Schlüssel* wird benötigt, um die Kennungen der Leistungserbringer zu pseudonymisieren.

Kommandozeilenoption: `-cn` oder `--certnew`

Über diese Option wird der Pfad zur neuen Zertifikatsdatei angegeben. Der Wert kann ein absoluter oder ein relativer Pfad sein. Zum Zeitpunkt der Verarbeitung muss die verwendete neue Zertifikatsdatei gültig sein (aktuelles Datum zwischen Start- und Enddatum des Zertifikates).

4.1.9 Schlüsselpaarerzeugung

Diese Funktion dient dazu, ein Schlüsselpaar bestehend aus einem Zertifikat und einem privaten Schlüssel zu generieren. Dabei ist zu beachten:

- Nur die Datenannahmestellen der Krankenhäuser LKG und die Kassenärztlichen Vereinigungen (KV) bzw. Kassenzahnärztlichen Vereinigungen (KZV) dürfen öffentliche Zertifikate und zugehörige private Schlüssel generieren.
- Die Übermittlung des öffentlichen Zertifikates von LKG/KV/KZV zur DAS-KK über einen sicheren Kanal ist näher in der beiliegenden Datenflussspezifikation in Kapitel 3 "Datenflüsse" beschrieben.
- Der private Schlüssel darf nur vom Schlüsselersteller für die Depseudonymisierung (Entschlüsselung des Langpseudonyms) verwendet werden.

Die Ausgabe des Programmes besteht aus zwei Dateien, die von der Anwendung im Ausgabeverzeichnis erzeugt werden:

- Das Zertifikat <dasType>_<land>_<validFrom_yyyyMMdd>-<validTo_yyyyMMdd>.cer
- Die Keystore-Datei, die das Schlüsselpaar und insbesondere den privaten Schlüssel enthält: <dasType>_<land>_<validFrom_yyyyMMdd>-<validTo_yyyyMMdd>.p12

Der Keystore und der Keystore-Eintrag sind jeweils mit dem übergebenen Passwort geschützt.

Funktionsparameter: `k` oder `keys`

Typ der Datenannahmestelle

Mögliche Werte als Typ der Datenannahmestelle: LQS oder KV.

Kommandozeilenoption: `-d` oder `--dastype`

Die Landesangabe

Das Land wird durch einen zweistelligen Länderkode angegeben. Zulässig sind die folgenden 18 Länderkennungen: BA, BB, BE, BW, HB, HE, HH, MV, NI, NO, NW, RP, SH, SL, SN, ST, TH, WL.

Kommandozeilenoption: `-l` oder `--land`

Das Von-Datum des Gültigkeitszeitraumes

Für das Zertifikat, das erzeugt werden soll, wird ein Gültigkeitszeitraum angegeben. Ein Bestandteil ist das Datum, ab dem das Zertifikat gültig sein soll. Das Datum wird in der Form dd.MM.yyyy angegeben, also z.B. 31.03.2023.

Kommandozeilenoption: `-f` oder `--from`

Das Bis-Datum des Gültigkeitszeitraumes

Der zweite Bestandteil des Gültigkeitszeitraums des Zertifikates ist das Bis-Datum. Es wird genauso wie das Von-Datum in der Form dd.MM.yyyy angegeben also z.B. 13.08.2027. Empfohlen wird die Angabe eines Gültigkeitszeitraums mit einer Dauer zwischen einem und fünf Jahren. Wird eine davon abweichende Gültigkeitsdauer angegeben, erfolgt eine Warnung.

Kommandozeilenoption: `-t` oder `--to`

Das Passwort

Der erzeugte Keystore (p12-Datei) wird als Keystore-Datei ins Dateisystem geschrieben. Diese Keystore-Datei liegt im PKCS#12 Format vor. Dabei handelt es sich in der Kryptographie um ein Dateiformat für ein Archiv, in welchem kryptographische Objekte abgelegt werden können. (Siehe https://de.wikipedia.org/wiki/Public-Key_Cryptography_Standards). In diesem Keystore wird für den generierten privaten Schlüssel ein entsprechender Eintrag angelegt. Der Keystore selbst und der einzelne Schlüsseleintrag werden mit einem Passwort verschlüsselt. Dieses Passwort wird über den folgenden Kommandozeilenparameter spezifiziert. Es muss den folgenden Kriterien entsprechen:

- mindestens 8 Zeichen lang
- mindestens ein Grossbuchstabe muss enthalten sein
- mindestens eine Zahl muss enthalten sein
- Es besteht ansonsten aus den Zeichen: Kleinbuchstaben a-z, Großbuchstaben A-Z, Ziffern 0-9 oder den Zeichen .,:;=+*#_!$%?@(){}|'^~

(Insbesondere sind Umlaute und andere Sonderzeichen wie z.B. § nicht erlaubt.)

Kommandozeilenoption: `-p` oder `--password`

Das Ausgabeverzeichnis

Über diesen Kommandozeilenparameter wird der Pfad des Verzeichnisses angegeben, in dem die Zertifikatsdatei (.cer) und die Keystoredatei (.p12) von der Anwendung erzeugt werden sollen. Das Verzeichnis muss bereits existieren.

Kommandozeilenoption: `-o` oder `--outdir`

4.1.10 Rückgabewerte und Fehlerbehandlung

Die Anwendung erzeugt System-Return-Codes, um Erfolgs- oder Fehlersituationen zu kommunizieren. Folgende Rückgabewerte werden unterschieden:

- **Rückgabewert gleich 0**: Die Abarbeitung war erfolgreich.
- **Rückgabewert kleiner 0:** Es ist ein technischer Fehler aufgetreten. In diesem Fall wird eine Logdatei relativ zum aktuellen Verzeichnis unter `logs\cli_public_error.log` abgelegt. Diese kann der BAS über das Kontaktformular https://iqtig.org/kontakt/verfahrenssupport zur weiteren Analyse zur Verfügung gestellt werden.
- **Rückgabewert größer 0:** Es ist ein fachlicher Fehler aufgetreten, der durch eine angepasste Eingabe des Anwenders behoben werden kann.

4.2 Programmierschnittstelle (API)

Die Funktionalität des PSP kann auch über die Java-API aufgerufen werden. API ist die Abkürzung von Application Programming Interface und meint im Falle einer Java-API eine Menge von Java-Klassen/Interfaces, über die die Funktionen des Programms zur Verfügung gestellt werden. Zum Ausführen bestimmter Funktionen müssen entsprechende Java-Methoden, die Teil der API sind, aufgerufen werden. Die verfügbaren Methoden werden in diesem Kapitel vorgestellt.

Es existieren drei Interfaces, welche die API repräsentieren und im Folgenden näher erklärt werden:

- org.iqtig.od.psp.api.KeyManagement
- org.iqtig.od.psp.api.Pseudonymisator
- org.iqtig.od.psp.api.Depseudonymisator

4.2.1 Interface KeyManagement

Das Interface KeyManagement enthält die folgenden Methoden:

Erzeugung eines Zertifikates mit privatem Schlüssel

```
CertificateWithPrivateKeyTO generateCertificateWithKeyPair(
        CertificateCreationTO certificateCreationTO
) throws PspApiException;
```

Diese Methode dient der Erzeugung eines RSA-Schlüsselpaares bestehend aus einem x509-Zertifikat und dem zugehörigen privaten Schlüssel.

- Eingabe: Das Objekt *CertificateCreationTO* enthält die für die Erzeugung des Zertifikates notwendigen Informationen:
  - *DataCollectionPoint*: Die Datenannahmestelle, für die mit diesem Zertifikat pseudonymisiert werden soll
  - *Region*: Region der Datenannahme. Diese Regionen umfassen alle Bundesländer sowie speziell unter Nordrhein-Westfalen die Regionen Westfalen-Lippe und Nordrhein.
  - *startDate*: Gültigkeitsbeginn des zu erzeugenden Zertifikates
  - *endDate*: Gültigkeitsende des zu erzeugenden Zertifikates
- Ausgabe: Das zurückgegebene Objekt *CertificateWithPrivateKeyTO* enthält die folgenden Attribute:
  - *x509Certificate*: Das erzeugte Zertifikat
  - *privateKey*: Der zum Zertifikat gehörende private Schlüssel

Erzeugung eines Zertifikates mit privatem Schlüssel ausgegeben als PKCS12-Datei

```
void generateCertificateWithKeyPairAndStoreToKeyStore(
        CertificateCreationTO certificateCreationTO,
        String password,
        OutputStream keyStoreOutputStream
) throws PspApiException;
```

Diese Methode erzeugt ein RSA-Schlüsselpaar, bestehend aus einem x509-Zertifikat und dem zugehörigen privaten Schlüssel und speichert beide Elemente in einem PKCS12-Keystore-OutputStream. Dabei wird der Keystore mit dem übergebenen Passwort verschlüsselt und der Schlüsseleintrag im Keystore wird ebenfalls mit dem Passwort verschlüsselt.

- Eingabe:
  - *certificateCreationTO*: Enthält gleiche Informationen wie das entsprechende Eingabe-Objekt in Interface-Methode *Erzeugung eines Zertifikates mit privatem Schlüssel*.
  - *password*: Passwort, mit dem Keystore und Keyentry verschlüsselt werden.
- Ausgabe:
  - *keyStoreOutputStream*: Über diesen Outputstream wird die PKCS12-Datei ausgegeben. Es bietet sich an, eine Datei mit der Dateiendung ".p12" zu erzeugen.

Extraktion eines Zertifikates aus PKCS12-Datei und Ausgabe in Outputstream

```
void extractAndStoreX509CertificateFromKeyStore(
        InputStream keyStoreInputStream,
        String password,
        OutputStream certificateOutputStream
) throws PspApiException;
```

Diese Methode extrahiert aus dem übergebenen Keystore-Inputstream das x509-Zertifikat und gibt es über den übergebenen Zertifikats-Outputstream aus. (Die Extraktion ist nur möglich, wenn das Keystore-Passwort und das Keyentry-Passwort übereinstimmt, was immer der Fall ist, wenn das Zertifikat über das Keymanagement-Interface erzeugt worden ist.) Beim Auslesen wird erwartet, dass das Schlüsselpaar im Keystore unter dem Alias "Pseudonymisierung" abgelegt ist, was auch sichergestellt ist, wenn das Zertifikat über das Keymanagement-Interface erzeugt wurde.

- Eingabe:
  - *keyStoreInputStream*: InputStream, der den Keystore enthält. In dem Keystore muss das zu extrahierende Zertifikat vorhanden sein.
  - *password*: Passwort zum Entsperren von Keystore und Keyentry.
- Ausgabe:
  - *certificateOutputStream*: Über diesen Outputstream wird das extrahierte Zertifikat ausgegeben.

Extraktion eines Zertifikates aus PKCS12-Datei

```
CertificateWithPrivateKeyTO readCertificateWithKeyPairFromKeyStore(
        InputStream keyStoreInputStream,
        String password
) throws PspApiException;
```

Diese Methode extrahiert aus dem übergebenen Keystore-Inputstream das x509-Zertifikat und den zugehörigen privaten Schlüssel und gibt beide Elemente als Java-Objekt *CertificateWithPrivateKeyTO* zurück. (Die Extraktion ist nur möglich, wenn das Keystore-Passwort und das Keyentry-Passwort übereinstimmt, was immer der Fall ist, wenn das Zertifikat über das Keymanagement-Interface erzeugt worden ist.) Beim Auslesen wird erwartet, dass das Schlüsselpaar im Keystore unter dem Alias "Pseudonymisierung" abgelegt ist, was sichergestellt ist, wenn das Zertifikat über das Keymanagement-Interface erzeugt wurde.

- Eingabe:
  - *keyStoreInputStream*: InputStream, der den Keystore enthält. In dem Keystore muss das zu extrahierende Schlüsselpaar unter dem Alias "Pseudonymisierung" vorhanden sein.
  - *password*: Passwort zum Entsperren von Keystore und Keyentry.
- Ausgabe: Das zurückgegebene Objekt *CertificateWithPrivateKeyTO* enthält die folgenden Attribute:
  - *x509Certificate*: Das extrahierte Zertifikat
  - *privateKey*: Der zum Zertifikat gehörende private Schlüssel

Einlesen eines x509-Zertifikates aus Schlüsseldatei im PEM-Format

```
X509Certificate readCertificateFromPem(InputStream certificateInputStream)
throws PspApiException;
```

Diese Methode liest ein Zertifikat ein und gibt es als Java-Objekt *X509Certificate* zurück. Die Methode erwartet, dass das Zertifikat im PEM-Format vorliegt.

- Eingabe:
  - *certificateInputStream*: InputStream, der das einzulesende x509-Zertifikat im PEM-Format enthält.
- Ausgabe: Das zurückgegebene Objekt *X509Certificate* enthält das eingelesene Zertifikat.

4.2.2 Interface Pseudonymisator

Pseudonymisieren einer IK- oder BSNR entsprechend dem übergebenen Pseudonymisierungsverfahren

```
String pseudonymiseIKBS(
        String ikBs,
        PseudonymisationProcedure pseudonymisationProcedure
) throws PspApiException;
```

Diese Methode pseudonymisiert eine IK- oder BSNR mit dem übergebenen Pseudonymisierungsverfahren.

- Eingabe:
  - *ikBs*: Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers, welche/welches pseudonymisiert werden soll.
  - *pseudonymisationProcedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll. Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) verwendet werden.
- Ausgabe: Der zurückgegebene String enthält das erzeugte Pseudonym.

Pseudonymisieren einer StandortId entsprechend dem übergebenen Pseudonymisierungsverfahren

```
String pseudonymiseStoId(
        String stoId,
        PseudonymisationProcedure pseudonymisationProcedure
) throws PspApiException;
```

Diese Methode pseudonymisiert eine StandortId mit dem übergebenen Pseudonymisierungsverfahren.

- Eingabe:
  - *stoId*: StandortId, die pseudonymisiert werden soll.
  - *pseudonymisationProcedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll. Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) verwendet werden.
- Ausgabe: Der zurückgegebene String enthält das erzeugte Pseudonym.

Pseudonymisieren einer lebenslangen Arztnummer entsprechend dem übergebenen Pseudonymisierungsverfahren

```
String pseudonymiseLAnr(
        String lAnr,
        PseudonymisationProcedure pseudonymisationProcedure
) throws PspApiException;
```

Diese Methode pseudonymisiert eine lebenslange Arztnummer mit dem übergebenen Pseudonymisierungsverfahren.

- Eingabe:
  - *lAnr*: lebenslange Arztnummer, die pseudonymisiert werden soll.
  - *pseudonymisationProcedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll. Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) verwendet werden.
- Ausgabe: Der zurückgegebene String enthält das erzeugte Pseudonym.

Pseudonymisieren einer Liste von IkBs/StoId-Kennungen

```
List<PseudonymOutputTO> pseudonymiseList(List<PseudonymInputTO> input)
throws PspApiException;
```

Diese Methode pseudonymisiert eine übergebene Liste von IkBs/StoId-Kennungen.

- Eingabe: Ein Listenelement besitzt den Typ *PseudonymInputTO*, dieser enthält:
  - *procedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert wird. Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) verwendet werden.
  - *ikBs*: Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers
  - *stoId*: Die StandortId

In jedem Listenelement muss mindestens eines der beiden Merkmale IkBs bzw. StoId gefüllt sein.

- Ausgabe: Zu jedem Eingabe-Listenelement vom Typ *PseudonymInputTO* wird ein Ausgabe-Listenelement vom Typ *PseudonymOutputTO* erzeugt, das zu den erzeugten Pseudonymen auch die Information des entsprechenden Eingabe-Listenelementes enthält:
  - *procedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll.
  - *ikBs*: Die Betriebsstättennummer oder das Institutskennzeichen des Leistungserbringers.
  - *stoId*: Die StandortId
  - *ikBsPseudonym*: Das aus der iKBs erzeugte Pseudonym gemäß dem Pseudonymisierungsverfahren in *procedure*
  - *stoIdPseudonym*: Das aus der StandortId erzeugte Pseudonym gemäß dem Pseudonymisier-

ungsverfahren in *procedure*

Pseudonymisieren einer Liste von lAnr-Kennungen

```
List<PseudonymLAnrOutputTO> pseudonymiseLAnrList(
        List<PseudonymLAnrInputTO> input
) throws PspApiException;
```

Diese Methode pseudonymisiert eine übergebene Liste von lAnr-Kennungen.

- Eingabe: Ein Listenelement besitzt den Typ *PseudonymLAnrInputTO*, dieser enthält:
  - *procedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll. Hier können alle zur DeQS-RL gehörenden Pseudonymisierungsverfahren sowie die Pseudonymisierungsverfahren DK (Darmkrebs) und ZK (Zervixkarzinom) aus der oKFE-RL (Richtlinie für organisierte Krebsfrüherkennungsprogramme) verwendet werden.
  - *lAnr*: Die lebenslange Arztnummer.

In jedem Listenelement müssen die beiden Merkmale Pseudonymisierungsverfahren und lAnr gefüllt sein.

- Ausgabe: Zu jedem Eingabe-Listenelement vom Typ *PseudonymLAnrInputTO* wird ein Ausgabe-Listenelement vom Typ *PseudonymLAnrOutputTO* erzeugt, das zu den erzeugten Pseudonymen auch nochmal die Informationen des entsprechenden Eingabe-Listenelementes enthält:
  - *procedure*: Pseudonymisierungsverfahren, gemäß dem pseudonymisiert werden soll.
  - *lAnr*: die lebenslange Arztnummer.
  - *lAnrPseudonym*: Das aus der lAnr erzeugte Pseudonym gemäß dem Pseudonymisierungsverfahren in *procedure*

4.2.3 Interface Depseudonymisator

Depseudonymisierung eines Pseudonyms

```
DepseudonymTO depseudonymise(String pseudonym) throws PspApiException;
```

Diese Methode depseudonymisiert ein Pseudonym und gibt das erhaltene Klartext-Objekt zusammen mit dem Pseudonymisierungsverfahren zurück.

- Eingabe:
  - *pseudonym*: Das zu depseudonymisierende Pseudonym
- Ausgabe: Das zurückgegebene Java-Objekt *DepseudonymTO* enthält:
  - *identifier*: StoId- bzw. IkBs-Kennung im Klartext
  - *pseudonymisationProcedure*: Pseudonymisierungsverfahren (ist in Pseudonym enthalten)

Depseudonymisierung einer Liste von IkBs- bzw. StoId-Pseudonymen

```
List<DepseudoOutputTO> depseudonymiseList(List<DepseudoInputTO> pseudonymList)
throws PspApiException;
```

Diese Methode depseudonymisiert eine Liste von IkBs- bzw. StoId-Pseudonymen und gibt eine Liste von *DepseudoOutputTO*-Objekten zurück, die unter anderem die depseudonymisierten Klartext-Objekte enthalten.

- Eingabe: Ein *DepseudoInputTO*-Objekt enthält:
  - *ikBsPseudonym*: IkBs-Pseudonym.
  - *stoIdPseudonym*: StoId-Pseudonym.

Mindestens eines der Pseudonym-Attribute *ikBsPseudonym* bzw. *stoIdPseudonym* muss gefüllt sein.

- Ausgabe: Zu jedem Eingabe-Listenelement wird ein Ausgabe-Listenelement erzeugt, das auch die Informationen des Eingabe-Listenelements enthält:
  - *procedure*: Das Pseudonymisierungsverfahren, gemäß dem die IkBs bzw. StoId pseudonymisiert worden sind. Sollten sie nicht übereinstimmen, wird eine Warnung ausgegeben und für das Attribut das Verfahren des IkBs-Pseudonyms übernommen.
  - *ikBs*: Die IkBs-Kennung im Klartext
  - *stoId*: Die StoId-Kennung im Klartext
  - *ikBsPseudonym*: IkBs-Pseudonym des entsprechenden Eingabe-Listenelementes
  - *stoIdPseudonym*: StoId-Pseudonym des entsprechenden Eingabe-Listenelementes

Depseudonymisierung einer Liste von lAnr-Pseudonymen

```
List<DepseudoLAnrOutputTO> depseudonymiseLAnrList(
        List<DepseudoLAnrInputTO> pseudonymList
) throws PspApiException;
```

Diese Methode depseudonymisiert eine Liste von lAnr-Pseudonymen und gibt eine Liste von *DepseudoLAnrOutputTO*-Objekten zurück, die unter anderem das depseudonymisierte Klartext-Objekt enthalten.

- Eingabe: Ein *DepseudoLAnrInputTO*-Objekt enthält:
  - *lAnrPseudonym*: lAnr-Pseudonym.

Das Pseudonym-Attribut *lAnrPseudonym* muss gefüllt sein.

- Ausgabe: Zu jedem Eingabe-Listenelement wird ein Ausgabe-Listenelement erzeugt, das auch die Informationen des Eingabe-Listenelementes enthält:
  - *procedure*: Das Pseudonymisierungsverfahren, gemäß dem die lAnr pseudonymisiert worden sind.
  - *lAnr*: Die lAnr-Kennung im Klartext
  - *lAnrPseudonym*: lAnr-Pseudonym des entsprechenden Eingabe-Listenelementes

4.2.4 API einzeln einbinden

Alternativ zum vollumfänglichen CLI - JAR lassen sich die Programmbibliotheken für die API und die API-Implementierung auch einzeln einbinden, falls im eigenen Projekt die ausschließliche Nutzung des PSPs als API ohne Gebrauch der CLI - Funktionalitäten gewünscht ist. Im Ordner "API" finden sich die einzelnen JAR-Pakete für die API / API-Implementierung und die passenden Maven-POMs mit Angabe der Abhängigkeiten. Die JAR-Dateien lassen sich im lokalen Maven-Repository oder auch direkt im Projektordner einbinden, allerdings sind die von den Bibliotheken benötigten externen Abhängigkeiten / Dependencies absichtlich nicht Bestandteil der JARs, damit sie platzsparender bzw. nicht-redundant im eigenen Projekt gepflegt werden können. Die mitgelieferten POM-Dateien sollen bei der Einbindung dieser (transitiven) Abhängigkeiten helfen, die dann im Projekt über Maven / Gradle etc. verwaltet werden können.

Die Maven-Artefakte können folgendermaßen im lokalen Repository installiert werden, vom entpackten API-Ordner aus:

- Haupt-POM

```
...\API> mvn install:install-file -Dpackaging="pom" -Dfile="psp-4.0.8.pom"
-DpomFile="psp-4.0.8.pom"
```

- API-JAR

```
...\API> mvn install:install-file -Dpackaging="jar" -Dfile="psp-api-4.0.8.jar"
-DpomFile="psp-api-4.0.8.pom"
```

- IMPL-JAR

```
...\API> mvn install:install-file -Dpackaging="jar" -Dfile="psp-impl-4.0.8.jar"
-DpomFile="psp-impl-4.0.8.pom"
```

Im Ordner "API\Beispiel" befindet sich ein Maven-Beispielprojekt, bei dem die API und API-Impl-Module per Maven eingebunden werden. In den JUnit-Tests werden exemplarisch einzelne API-Funktionen aufgerufen.

5 Migration

5.1 Migration der API von Version 2.x zu Version 4.x

Die API des PSPs in Version 2.x besteht aus folgenden Interfaces:

- CertificateStore
- CsvDepseudonymisator
- CsvPseudonymisator
- IkbsAndStoDepseudonymisator
- IkbsAndStoPseudonymisator
- Registration
- StringDepseudonymisator
- StringPseudonymisator
- X509CertificateGenerator

Im Folgenden ist je ein Kapitel einem der entsprechenden Interfaces gewidmet. In den Kapiteln wird erläutert, welche Methoden im PSP in Version 2.x existieren und wie die Funktionalität durch die API des PSP in Version 4.x, die im Kapitel 4.2 beschrieben wurde, abgebildet wird.

5.1.1 Interface CertificateStore

Für die Funktionalität dieses Interfaces gibt es in der aktuellen PSP-API keine Entsprechung.

5.1.2 Interface CsvDepseudonymisator

Für die Funktionalität dieses Interfaces gibt es in der aktuellen PSP-API keine direkte Entsprechung. Mit der Methode "depseudonymiseList" bzw. "depseudonymiseLAnrList" des Interfaces "Depseudonymisator" kann jedoch eine Liste von IkBs-, StoId- bzw. IAnr-Pseudonymen depseudonymisiert werden.

5.1.3 Interface CsvPseudonymisator

Für die Funktionalität dieses Interfaces gibt es in der aktuellen PSP-API keine direkte Entsprechung. Mit der Methode "pseudonymiseList" bzw. "pseudonymiseLAnrList" des Interfaces "Pseudonymisator" kann jedoch eine Liste von IkBs-, StoId- bzw. IAnr-Kennungen pseudonymisiert werden.

5.1.4 Interface IkbsAndStoDepseudonymisator

Methode zur Depseudonymisierung eines Strings

```
PseudonymTO depseudonymise(String pseudonym) throws PseudonymisationException;
```

Diese Methode depseudonymisiert einen String und gibt das Ergebnis samt ermitteltem Pseudonymisierungsverfahren in dem Rückgabeobjekt zurück.

- Eingabe: *pseudonym*, der zu depseudonymisierende String.
- Ausgabe: Das zurückgegebene Java-Objekt *PseudonymTO* enthält:
  - *leIdentifer*: Der Klartext-String als Ergebnis der Depseudonymisierung
  - *pseudonymisierungsVerfahren*: Das durch die Depseudonymisierung ermittelte Pseudonymisierungsverfahren
- Entsprechende Funktion in neuer API: Die Funktion wird in 4.2.3 Interface Depseudonymisator durch die Methode *Depseudonymisierung eines Pseudonyms* abgedeckt.

Depseudonymisierung einer Liste von Pseudonymen

```
List<PseudonymTO> depseudonymiseList(List<String> pseudonyms)
throws PseudonymisationException;
```

Diese Methode depseudonymisiert eine Liste übergebener Strings und gibt jeweils den Klartext-String und das Pseudonymisierungsverfahren im jeweiligen Rückgabeobjekt als Liste zurück.

- Eingabe: *pseudonyms*, Liste der zu depseudonymisierenden Strings.
- Ausgabe: Eine Liste von Java-Objekten *PseudonymTO*, pro Eingabe-String wird ein *PseudonymTO* -Objekt zurückgegeben. Das enthält:
  - *leIdentifer*: Der Klartext-String als Ergebnis der Depseudonymisierung
  - *pseudonymisierungsVerfahren*: Das durch die Depseudonymisierung ermittelte Pseudonymisierungsverfahren
- Entsprechende Funktion in neuer API: Die Funktion wird in 4.2.3 Interface Depseudonymisator durch die Methoden *Depseudonymisierung einer Liste von IkBs- bzw. StoId-Pseudonymen* und *Depseudonymisierung einer Liste von IAnr-Pseudonymen* abgedeckt. Dabei gibt es zwei Fälle:
  - Es soll eine Liste von ikBs- oder stoId-Pseudonymen depseudonymisiert werden: Es ist die Methode unter *Depseudonymisierung einer Liste von IkBs- bzw. StoId-Pseudonymen* zu verwenden. Dabei muss die Liste von Strings auf eine Liste von Eingabe-Objekten *DepseudoInputTO* abgebildet werden, indem gewählt wird, ob das *ikBsPseudonym*-Attribut oder das *stoIdPseudonym*-Attribut des *DepseudoInputTO*-Objektes gesetzt wird. Dementsprechend enthält das Rückgabe-Objekt vom Typ *DepseudoOutputTO* den depseudonymisierten String im *ikBs*-Attribut oder im *stoId*-Attribut.
  - Es soll eine Liste von IAnr-Pseudonymen depseudonymisiert werden: Es ist die Methode unter *Depseudonymisierung einer Liste von IAnr-Pseudonymen* zu verwenden. Dabei müssen die *IAnrPseudonym*-Attribute der *DepseudoLAnrInputTO*-Listenelemente entsprechend gesetzt werden.

5.1.5 Interface IkbsAndStoPseudonymisator

Methode zur Erzeugung eines IkBs-Pseudonyms

```
String createIkbsPseudonym(
        String ikbs,
        Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode erstellt zu einer Klartext-IkBs und einem Pseudonymisierungsverfahren ein IkBs-Pseudonym.

- Eingabe:
  - *ikbs*: Klartext-IkBs, die pseudonymisiert werden soll.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: String, der die pseudonymisierte IkBs darstellt
- Entsprechende Funktion in neuer API: 4.2.2 Interface Pseudonymisator Methode *Pseudonymisieren einer IK- oder BSNR entsprechend dem übergebenen Pseudonymisierungsverfahren*. Dabei muss statt der Klasse *org.iqtig.pseudonymisierung.enums.Method* die Klasse *org.iqtig.od.psp.api.enums.-PseudonymisationProcedure* zur Kodierung des Pseudonymisierungsverfahrens verwendet werden, ansonsten ist die Signatur dieser Methode bezüglich eingegebener IkBs und zurückgegebenem Pseudonym zur erwähnten Methode der neuen API analog.

Methode zur Erzeugung einer Liste von IkBs-Pseudonymen

```
List<String> createIkbsPseudonyms(
        List<String> ikbsList,
        Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode erstellt zu einer Liste von Klartext-IkBs und einem Pseudonymisierungsverfahren eine Liste IkBs-Pseudonymen.

- Eingabe:
  - *ikbsList*: Klartext-IkBs-Liste, deren Elemente pseudonymisiert werden sollen.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: Liste mit IkBs-Pseudonymen
- Entsprechende Funktion in neuer API: 4.2.2 Interface Pseudonymisator Methode *Pseudonymisieren einer Liste von IkBs/StoId-Kennungen*. Für jeden zu pseudonymisierenden String muss ein Java-Objekt vom Typ *PseudonymInputTO* erstellt werden, in dem das Pseudonymisierungsverfahren im Attribut *procedure* und der zu pseudonymisierende String im Attribut *ikBs* gesetzt wird. Diese Liste von *PseudonymInputTO*-Objekten wird dann der Methode übergeben. Zu jedem *PseudonymInputTO* -Inputobjekt erzeugt die Methode ein *PseudonymOutputTO*-Outputobjekt. Darin ist jeweils im Attribut *ikBsPseudonym* dann das erzeugte Pseudonym enthalten.

Methode zur Erzeugung eines StoId-Pseudonyms

```
String createStoPseudonym(
        String stoId,
        Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode erstellt zu einer Klartext-StoId und einem Pseudonymisierungsverfahren ein StoId-Pseudonym.

- Eingabe:
  - *stoId*: Klartext-StoId, die pseudonymisiert werden soll.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: String, der die pseudonymisierte StoId darstellt
- Entsprechende Funktion in neuer API: 4.2.2 Interface Pseudonymisator Methode *Pseudonymisieren einer StandortId entsprechend dem übergebenen Pseudonymisierungsverfahren*. Dabei muss statt der Klasse *org.iqtig.pseudonymisierung.enums.Method* die Klasse *org.iqtig.od.psp.api.enums.-PseudonymisationProcedure* zur Kodierung des Pseudonymisierungsverfahrens verwendet werden, ansonsten ist die Signatur bezüglich eingegebener StoId und zurückgegebenem Pseudonym analog.

Methode zur Erzeugung einer Liste von StoId-Pseudonymen

```
List<String> createStoPseudonyms(
        List<String> stoIds,
        Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode erstellt zu einer Liste von Klartext-StoIds und einem Pseudonymisierungsverfahren eine Liste StoId-Pseudonyme.

- Eingabe:
  - *stoIds*: Klartext-StoId-Liste, deren Elemente pseudonymisiert werden.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: Liste mit pseudonymisierten StoIds
- Entsprechende Funktion in neuer API: 4.2.2 Interface Pseudonymisator Methode *Pseudonymisieren einer Liste von IkBs/StoId-Kennungen*. Für jeden zu pseudonymisierenden String muss ein Java-Objekt vom Typ *PseudonymInputTO* erstellt werden, in dem das Pseudonymisierungsverfahren im Attribut *procedure* und der zu pseudonymisierende String im Attribut *stoId* gesetzt wird. Diese Liste von *PseudonymInputTO*-Objekten wird dann der Methode übergeben. Zu jedem *PseudonymInputTO* -Inputobjekt erzeugt die Methode ein *PseudonymOutputTO*-Outputobjekt. Darin ist jeweils im Attribut *stoIdPseudonym* das erzeugte Pseudonym enthalten.

Methode zur Erzeugung eines lAnr-Pseudonyms

```
String createLanrPseudonym(
         String lanr,
         Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode generiert zur übergebenen lebenslangen Arztnummer und dem Pseudonymisierungsverfahren ein entsprechendes Pseudonym und gib es zurück.

- Eingabe:
  - *lanr*: lebenslange Arztnummer im Klartext.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: pseudonymisierte lebenslange Arztnummer.

Entsprechende Funktion in neuer API: 4.2.2 Interface Pseudonymisator *Pseudonymisieren einer lebenslangen Arztnummer entsprechend dem übergebenen Pseudonymisierungsverfahren*. Die Signatur ist analog zu verwenden.

Methode zur Erzeugung einer Liste von lAnr-Pseudonymen

```
List<String> createLanrPseudonyms(
         List<String> lanrs,
         Method pseudonymisierungsVerfahren
) throws PseudonymisationException;
```

Methode generiert zu einer Liste von lebenslangen Arztnummern eine Liste von Pseudonymen und gibt diese zurück.

- Eingabe:
  - *lanrs*: Liste von lebenslangen Arztnummern im Klartext.
  - *pseudonymisierungsVerfahren*: Pseudonymisierungsverfahren, mit dem pseudonymisiert werden soll.
- Ausgabe: Liste von pseudonymisierten lebenslangen Arztnummern.

Entsprechende Funktion in neuer API 4.2.2 Interface Pseudonymisator *Pseudonymisieren einer Liste von lAnr-Kennungen*. Diese Funktion der neuen API entspricht weitgehend der Funktion der alten API, wobei bei der neuen Funktion in jedem Eingabelelement vom Typ *PseudonymLAnrInputTO* nicht nur die zu pseudonymisierende lAnr angegeben werden muss, sondern auch das Pseudonymisierungsverfahren, während bei der alten Funktion das Pseudonymisierungsverfahren "global" für die gesamte Liste der lAnr-Kennungen angegeben werden musste. Das bedeutet insbesondere, dass sich bei der neuen Funktion das Pseudonymisierungsverfahren für jede übergebene lebenslange Arztnummer unterscheiden kann, ansonsten funktionieren beide Methoden analog.

5.1.6 Interface Registration

Für die Funktionalität dieses Interfaces gibt es im aktuellen PSP keine Entsprechung.

5.1.7 Interface StringDepseudonymisator

Für die Funktionalität dieses Interfaces gibt es im aktuellen PSP keine Entsprechung.

5.1.8 Interface StringPseudonymisator

Für die Funktionalität dieses Interfaces gibt es im aktuellen PSP keine Entsprechung.

5.1.9 Interface X509CertificateGenerator

Migrationspfad anhand des Beispiels der Erzeugung eines Zertifikates mit privatem Schlüssel abgelegt in einem PKCS12-Keystore

Hier wird beispielhaft der Fall einer Zertifikatserzeugung mit dem Interface X509CertificateGenerator und der Methode

```
void certificateGeneration(
        final Date startDate,
        final Date endDate
) throws PseudonymisationException;
```

auf die API des neuen PSP übertragen. Dabei wird ebenfalls davon ausgegangen, dass die verwendete Implementierung des Interfaces mit dem Konstruktor

```
public X509CertificateGeneratorImpl(
        final DataCollectionPoint dataCollectionPoint,
        final StateCode stateCode,
        final Path registrationDir,
        final String password
) throws PseudonymisationException;
```

erzeugt wurde. Bei diesem Anwendungsfall wird ein Zertifikat und ein privater Schlüssel erzeugt. Auf die API des neuen PSP überträgt man ihn wie folgt: Zunächst erzeugt man mit dem *APIBuilder* eine Instanz der Implementierung des Keymanagement-Interfaces. Daraufhin ruft man die Methode *Erzeugung von Zertifikat mit privatem Schlüssel ausgegeben in PKCS12-Datei* unter 4.2.1 Interface KeyManagement auf. Diese besitzt die Signatur

```
void generateCertificateWithKeyPairAndStoreToKeyStore(
        CertificateCreationTO certificateCreationTO,
        String password,
        final OutputStream keyStoreOutputStream
) throws PspApiException;
```

Dabei wird das Objekt *certificateCreationTO* mit den Informationen bestückt, die auch bei den Aufrufen (siehe oben Konstruktor + Methodenaufruf) der API des alten PSP eingegeben wurden:

- *startDate*
- *endDate*
- *dataCollectionPoint*
- *region* (entspricht *stateCode*)

Für den Parameter *password* in der neuen API-Methode wird der Inhalt von *password* des alten Aufrufes genommen.

Statt einem *registrationDir* wird bei dem neuen Aufruf der OutputStream *keyStoreOutputStream* übergeben, über den der erzeugte PKCS12-Keystore ausgegeben wird. Um bei der neuen API das Zertifikat als *cer*-Datei zu erhalten, muss zusätzlich die Methode *Extraktion von Zertifikat aus PKCS12-Datei* des Interfaces 4.2.1 Interface KeyManagement aufgerufen werden. Sie besitzt die Signatur

```
void extractAndStoreX509CertificateFromKeyStore(
         InputStream keyStoreInputStream,
         String password,
         OutputStream certificateOutputStream
) throws PspApiException;
```

Für den Parameter *keyStoreInputStream* wird nun ein Inputstream eingesetzt, der auf die im vorherigen Schritt erzeugte Keystore-Datei verweist. Für das *password* wird das im vorherigen Schritt verwendete Passwort genommen. Über den Parameter *certificateOutputStream* wird der Outputstream spezifiziert, über den das Zertifikat ausgegeben wird.

Bemerkung: In der alten API ist die Möglichkeit vorgesehen, das Datum für das Ende des Gültigkeitsraumes des Zertifikates wegzulassen. In diesem Fall wird als Ende des Gültigkeitszeitraumes das Datum für den Gültigkeitsbeginn des Zertifikates genommen plus zwanzig Jahre. Diese Möglichkeit ist in der neuen API nicht vorgesehen. Hier müssen stets Gültigkeitsbeginn und Gültigkeitsende angegeben werden.

5.2 Migration des CLI von Version 2.x zu Version 4.x

Für die Migration der Kommandozeilen-Schnittstelle des alten PSP zur Kommandozeilen-Schnittstelle des neuen PSP werden folgende Anwendungsfälle betrachtet:

1. Generierung von Zertifikaten
2. Pseudonymisierung von CSV zu CSV
3. Depseudonymisierung von CSV zu CSV
4. Übermittlung der Zusatzdaten an die BAS

5.2.1 Generierung von Zertifikaten

Die Erzeugung eines Zertifikates mit privatem Schlüssel mit den folgenden Parametern

- Datenannahmestelle: LQS
- Beginn der Gültigkeit des Zertifikates: 30.06.2015
- Ausgabe-Verzeichnis: c:\pseudo\key
- Land: BW
- Keystore- und Keyentry-Passwort: A2345678

wurde unter dem CLI des alten PSP wie folgt ausgelöst:

```
java -jar pseudonymisierung-console.jar
              -a LQS
              -b 30.06.2015
              -c C:\pseudo\key
              -l BW
              -s A2345678
              -z
```

Der äquivalente Aufruf mit dem neuen PSP lautet:

```
java -jar psp-cli.jar keys
              -d LQS
              -l BW
              -f 30.06.2015
              -t 30.06.2035
              -p A2345678
              -o C:\pseudo\key
```

Dabei ist zu beachten, dass beim neuen PSP zwingend auch ein Gültigkeitsende des Zertifikates mit Flag "t" spezifiziert werden muss, während beim alten PSP nur die Angabe eines Gültigkeitsbeginns verpflichtend war. Außerdem gelten für das Passwort nun Mindestanforderungen. (Siehe Kapitel 4.1.9 Schlüsselpaarerzeugung)

5.2.2 Pseudonymisierung von CSV zu CSV

Aufruf beim alten PSP-CLI:

```
java -jar pseudonymisierung-console.jar
              -a LQS
              -c C:\pseudo\key
              -l NI
              -i C:\pseudo\input.csv
              -j 2
              -k 1
              -p
              -r C:\pseudo\out\outk.csv
              -t "-1"
              -w PCI
```

Aufruf beim neuen PSP-CLI:

```
java -jar psp-cli.jar pseudonymisierung_csv
              -i C:\pseudo\input.csv
              -o C:\pseudo\out
              -c C:\pseudo\key\LQS$$NI20150630.cer
```

Unterschiede:

1. Datenannahmestelle, Land und Zertifikat:
    - Bei altem PSP-CLI wird die Datenannahmestelle (Flag "a") und das Land (Flag "l") explizit spezifiziert. Es wird außerdem mit Flag "c" ein Verzeichnis angegeben, aus dem ein Zertifikat geladen wird, das zur Datenannahmestelle und dem Land passt.
    - In dem neuen PSP-CLI wird dagegen die Zertifikatsdatei konkret spezifiziert. Die Informationen der Datenannahmestelle und des Landes werden aus diesem Zertifikat extrahiert.
2. Spalten- und Zeilenindizes
    - Bei dem alten PSP-CLI wird zum einen die Zeile angegeben (Flag "j"), ab der die Verarbeitung beginnen soll. Außerdem wird mit Flag "k" die Spalte angegeben, aus der die Kennung ausgelesen wird, die dann pseudonymisiert wird. Darüber hinaus kann hier über Flag "t" angegeben werden, in welche Spalte in der Ausgabe-CSV-Datei das Ergebnis der Pseudonymisierung geschrieben wird.
    - In dem neuen PSP-CLI wird eine feste Position und Reihenfolge der Spalten erwartet. Außerdem werden alle Zeilen verarbeitet. In der Ausgabedatei folgen nach den Eingabespalten die Spalten "ikbs_pseudonym" und "stoid_pseudonym" an festen Positionen, die dann nach Ausführung des Programms das Ergebnis der Pseudonymisierung enthalten. (Siehe Kapitel 4.1.3 Pseudonymisierung - Erstellung einer Liste von Pseudonymen aus einer Eingabeliste von Klartext-Kennungen)

3. Pseudonymisierungsverfahren
    - Bei dem alten PSP-CLI wird das Pseudonymisierungsverfahren über das Flag "w" spezifiziert.
    - Im neuen PSP-CLI wird das Pseudonymisierungsverfahren als erste Spalte im eingegebenen CSV erwartet und deren Werte können sich von Zeile zu Zeile unterscheiden.
4. Ausgabeverzeichnis: Während beim alten PSP-CLI ein konkreter Pfad mit Dateinamen für die Ausgabe-CSV-Datei angegeben wird (Flag "r"), gibt man beim neuen PSP-CLI mit Flag "o" nur ein Verzeichnis an. In diesem Verzeichnis wird dann die Ausgabedatei mit generiertem Dateinamen erzeugt.

5.2.3 Depseudonymisierung von CSV zu CSV

Aufruf bei altem PSP-CLI:

```
java -jar pseudonymisierung-console.jar
              -a LQS
              -c C:\pseudo\key
              -d
              -l NI
              -i C:\pseudo\depseu.csv
              -j 2
              -k 1
              -r C:\pseudo\out\outdepseuk.csv
              -t "-1"
              -s A2345678
```

Aufruf bei neuem PSP-CLI:

```
java -jar psp-cli.jar depseudonymisierung_csv
              -i C:\pseudo\depseu.csv
              -o C:\pseudo
              -k C:\pseudo\key\LQS$$NI20150630.p12
              -p A2345678
```

Unterschiede:

1. Datenannahmestelle, Land und Keystoredatei:
    - Bei dem alten PSP-CLI wird die Datenannahmestelle (Flag "a") und das Land (Flag "l") explizit spezifiziert. Es wird außerdem mit Flag "c" ein Verzeichnis angegeben, aus dem die Keystore-datei geladen wird, das zur Datenannahmestelle und dem Land passt.
    - In dem neuen PSP-CLI wird die Keystoredatei mit Flag "k" konkret spezifiziert.
2. Spalten- und Zeilenindizes
    - Bei dem alten PSP-CLI wird zum einen die Zeile angegeben (Flag "j"), ab der die Verarbeitung beginnen soll. Außerdem wird mit Flag "k" die Spalte angegeben, aus der das Pseudonym ausgelesen wird, das dann depseudonymisiert wird. Darüber hinaus kann über das Flag "t" angegeben werden, in welche Spalte in der Ausgabe-CSV-Datei das Ergebnis der Depseudo-

nymisierung geschrieben wird.

- In dem neuen PSP-CLI wird eine feste Position und Reihenfolge der Spalten erwartet. Außerdem werden alle Zeilen verarbeitet. In der Ausgabedatei folgen nach den Eingabespalten die Spalten "pseudonymisierungsverfahren", "ikbs_kennung" und "stoid_kennung" an festen Positionen, die das Ergebnis der Depseudonymisierung enthalten. (Siehe Kapitel 4.1.5 Depseudonymisierung - Depseudonymisierung einer Eingabeliste im CSV-Format von Pseudonymen mit Ausgabe als CSV-Datei)

3. Ausgabeverzeichnis: Während beim alten PSP-CLI ein konkreter Pfad mit Dateinamen für die Ausgabe-CSV-Datei angegeben wird (Flag "r"), gibt man beim neuen PSP-CLI mit Flag "o" nur ein Verzeichnis an. In diesem Verzeichnis wird dann die Ausgabedatei mit generiertem Dateinamen erzeugt.

5.2.4 Übermittlung der Zusatzdaten an die BAS

Die alte PSP-CLI verfügte über eine Funktion zur Übermittlung von Zusatzdaten an die BAS. Ein solcher Aufruf sah beispielsweise so aus:

```
java -jar pseudonymisierung-console.jar
             -h
             -a LQS
             -c C:\pseudo\key
             -l NI
             -I C:\pseudo\pseudonymMappings.csv
             -m C:\pseudo\zusatzDaten.csv
             -n user-name
             -q password
```

Für diese Funktion gibt es in der neuen PSP-CLI keine Entsprechung.

6 Hilfe und Support

Für Fragen, Anregungen und Fehlermeldungen wenden Sie sich bitte per Kontaktformular an unser Team Verfahrenssupport https://iqtig.org/kontakt/verfahrenssupport.

Herausgeber:

IQTIG – Institut für Qualitätssicherung und Transparenz im Gesundheitswesen
Katharina-Heinroth-Ufer 1
10787 Berlin

Telefon: (030) 58 58 26-0
Telefax: (030) 58 58 26-999

info@iqtig.org https://www.iqtig.org