



Kassenärztliche
Bundesvereinigung
Körperschaft des öffentlichen Rechts

XPM-LDK.KV

Handbuch Prüfmodul

[*KBV_ITA_AHEX_Handbuch_Pruefmodul
_LDK_KV*]

Dezernat Digitalisierung und IT

10623 Berlin, Herbert-Lewin-Platz 2

Kassenärztliche Bundesvereinigung

Version 1.0.3
Datum: 01.03.2018
Klassifizierung: Extern
Status: In Kraft

Version	Datum	Autor	Änderung	Begründung	Seite
1.0.3	01.03.2018	KBV	Einfügen einer Beschreibung der Java-Klasse „XPMEinstieg“		17
1.0.2	15.11.2017	KBV	Prüfung der Signatur für Muster 10 hinzugefügt		21
1.0.1	15.06.2017	KBV	Einfügung eines Hinweises bei der Beschreibung der Prüfungen Anpassung der Prüfung des Statusfeldes		18 20
1.0.0	15.05.2017	KBV	Initiale Erstellung		Alle

Inhaltsverzeichnis

1. EINLEITUNG	5
1.1 Begriffsklärung.....	5
1.2 Systemvoraussetzungen	6
2. VERZEICHNISSTRUKTUR	7
2.1 Ordner 'Bin'	7
2.2 Ordner 'Daten'.....	7
2.3 Ordner 'Doku'	7
2.4 Ordner 'Geprueft/Abgelehnt'	7
2.5 Ordner 'Geprueft/NichtAbgelehnt'	7
2.6 Ordner 'Konfig'	7
2.7 Ordner 'Listen'.....	7
2.8 Ordner 'Schema'	7
3. ANWENDUNG DES PRÜFMODULES	8
3.1 Technische Hinweise.....	8
3.2 Prüfmodi des Prüfmoduls	8
3.3 Konfigurationsdatei	8
3.3.1 Abschnitt: Allgemeiner Teil	9
3.3.2 Abschnitt: Eingabedateien	10
3.3.3 Abschnitt: Ausgabedateien.....	11
3.3.4 Umgebungsvariablen.....	12
3.4 Arbeiten im Kommandozeilenmodus.....	13
3.4.1 Übergabeparameter.....	13
3.4.2 Beispiele	14
4. HINWEISE ZU JAVA	16
4.1 Starten eines Java-Programms	16
4.2 Starten des Prüfmoduls aus einem Java-Programm.....	17
5. PRÜFREGELN	18

5.1 Allgemeine Prüfung bei Verwendung des Modus „Digitales Muster“ oder „Digitales Muster mit Verzeichnisprüfung“.....	18
5.1.1 Prüffälle für Muster 10	19
5.1.2 Prüffälle für Muster 10A.....	21

1. Einleitung

Dieses Dokument gibt einen Überblick über die Arbeitsweise und eine genaue Beschreibung zur Installation und Ausführung des neuen LDK Prüfmoduls. Das Prüfmodul LDK ist ein XPM (Prüfmodul) der KBV. Daher wird „XPM-LDK“ und „LDK-Prüfmodul“ im vorliegenden Dokument synonym verwendet.

Das Prüfmodul in der Version für Kassenärztliche Vereinigungen ist als Service für die Kassenärztlichen Vereinigungen konzipiert. Es kann auch von Softwareherstellern genutzt werden – eine Pflicht für den Einsatz durch die Softwarehersteller existiert nicht.

Das LDK-Prüfmodul in dieser Version ist ein Prüfprogramm für digitale Muster, um formelle Prüfungen auf dem digitalen Muster durchzuführen. Das Prüfmodul bietet dafür einen Prüfmodus an.

Als Ergebnis der Prüfung wird unter anderem ein Prüfprotokoll erstellt, das Informationen über den Zustand der Prüfdatei enthält. Als Ergebnis der Prüfung sind je nach Zustand der Prüfdatei folgende Statusmeldungen möglich: Ok, Warnung, Fehlerhaft, Hinweis oder Abbruch.

Das LDK-Prüfmodul ist auf allen Computersystemen lauffähig, für die die Java Laufzeitumgebung in der Version 1.7 oder höher verfügbar ist.

1.1 Begriffsklärung

- **XPM_LDK:** LDK steht für Labordatenkommunikation. Unter diesem Begriff sind der LDT3.0 und digitale Muster 10 bzw. 10A zusammengefasst. Das XPM_LDK steht als Synonym für das Prüfmodul der digitalen Muster 10 und 10 A sowie für den LDT 3.0 Datensatz.
- **XPM_LDK-Paket:** Ein Paket des LDK-Prüfmoduls bestehend aus folgenden Teilen :
 - XPM-Kernel (JAVA-Programm zum Interpretieren des LDK-Prüfprojektes)
 - LDK-Prüfprojekt (LDK spezifische Erweiterung des XPM-Kernels)
 - ReleaseNotes (Textdatei, die alle Informationen zur Version des Prüfmodulpaketes enthält)
 - GUI (Aufruf des Prüfmoduls als graphische Benutzeroberfläche)
 - Batchdateien und Shellskripte (Beispiele für den kommandozeilenorientierten Aufruf in verschiedenen Modi)
- **XPM-LDK.KV-Va.b.c:** Schnittstellenspezifisches Prüfmodulpaket für digitale Muster. Prüfmodul-Version a.b.c auch "das aktuelle LDK-Prüfmodul" genannt.
- **XPM_LDK-Prüfprojekt:** Tabellen und Codeelemente, die die schnittstellenspezifische Konfiguration enthalten; schnittstellenspezifischer Bestandteil des XPM_ LDK - Pakets.
- **XPM-Kernel:** Programm zum Interpretieren des XPM_LDK-Prüfprojektes, der Kernel ist ohne verfahrensspezifisches Prüfprojekt nicht verwendbar; XPM-allgemeiner Bestandteil des XPM_LDK-Pakets.
- **"Prüfmodul-Version":** Bezeichnet die Version des XPM_LDK-Paketes.

- **"das aktuelle LDK-Prüfmodul"**: Hiermit ist meist die neuste Version des XPM_LDK-Pakets gemeint.
- **"Gesamtpaket XPM-LDK.praxis-Va.b.c"**: Das Gesamtpaket enthält alle relevanten Dokumente und Software für die LDT-Schnittstelle und die digitalen Muster. Achtung! Die Versionsnummer des Gesamtpakets ist nicht zu verwechseln mit der Versionsnummer des technischen Handbuchs oder des Prüfmoduls.

1.2 Systemvoraussetzungen

XPM ist eine Applikation für ein 32bit-System.

Benötigt wird ein Computersystem, das leistungsmäßig mit einem IBM-kompatiblen PC, 256 MB Hauptspeicher und einem Pentium-Prozessor 500 oder höher vergleichbar ist.

Langsamere Prozessoren und weniger Hauptspeicher erhöhen stark die Laufzeit des Programms. XPM benötigt weniger als 10 MByte Festplattenplatz.

2. Verzeichnisstruktur

Die Verzeichnisstruktur des LDK-Prüfmoduls hat folgenden Aufbau:

2.1 Ordner 'Bin'

Dieser Ordner beinhaltet alle Java-Archive und binären Steuerdateien, die zur Ausführung des LDK-Pakets benötigt werden.

2.2 Ordner 'Daten'

Dieser Ordner enthält die Prüfdateien.

2.3 Ordner 'Doku'

Dieser Ordner enthält die Dokumentation zum XPM_LDK.

2.4 Ordner 'Geprueft/Abgelehnt'

Dieser Ordner dient als Ablage für geprüfte Dateien. Hier werden die Dateien abgelegt, die fehlerhaft sind.

2.5 Ordner 'Geprueft/NichtAbgelehnt'

Dieser Ordner dient als Ablage für geprüfte Dateien. Hier werden die Dateien abgelegt, die fehlerfrei sind.

2.6 Ordner 'Konfig'

Dieser Ordner enthält Konfigurationsdatei(en) im XML-Format.

2.7 Ordner 'Listen'

In diesem Ordner werden alle Ausgaben des Prüfmoduls generiert.

2.8 Ordner 'Schema'

In diesem Ordner befinden sich die XML-Schemadateien, die das Prüfformat definieren. Die Schemadateien (*.xsd) dürfen **nicht** verändert werden.

3. Anwendung des Prüfmoduls

XPM_LDK steht als einheitliches Werkzeug zur Prüfung möglichst vieler Eingangs- und Ausgangsdaten zur Verfügung. Dabei verarbeitet es je nach gewähltem Startskript digitale Muster einzeln oder Digitale Muster in einem Verzeichnis. XPM_LDK läuft in einer Java Laufzeitumgebung und kann somit auf allen Betriebssystemen eingesetzt werden, auf denen die Java-Laufzeitumgebung installiert ist.

Ein Prüfmodullauf beinhaltet dabei den Start des Moduls über eines der mitgelieferten Startskripts (.bat) oder über den Kommandozeilenauftruf mit entsprechendem Parameter. Danach werden die dem Prüfmodul übergebenen Dateien geprüft und ein Ergebnis ausgegeben. Die Konfiguration eines Prüfmodullaufs erfolgt mit Hilfe einer XML-Konfigurationsdatei. Nähere Informationen zur Konfigurationsdatei finden Sie in Abschnitt 3.3.

3.1 Technische Hinweise

XPM_LDK besteht im Kern aus einem Steuermodul (pruefprogramm.ldk.x.x) sowie einem Prüfungsmodul (pruefmodul-ldk-a.b.c.jar).

Zur Prüfung von digitalen Mustern prüft das Prüfmodul die Eingangsdaten gegen ein internes Regelwerk. Das Ergebnis einer Prüfung (Fehlermeldungen, Informationsmeldungen) sowie evtl. andere Ausgabedateien (Statistikmeldungen) werden in Dateien ausgegeben, deren Format vom Benutzer konfigurierbar ist.

3.2 Prüfmodi des Prüfmoduls

Das Prüfmodul LDK wird, wie eingangs erwähnt, mit verschiedenen Übertragungsparametern und Startskripten ausgeliefert. Die folgende Tabelle soll Aufschluss darüber geben, wie diese zusammenhängen. Die genaue Funktion der Übertragungsparameter wird in Kapitel 3.4.1 erläutert.

Prüfmodus	Übertragungsparameter	Skriptname
Digitales Muster	-c, -i, -f Optional: -e, -s, -m, -v, -h, -p	StartPruefungPDF_Inhalt.bat und StartPruefungPDF_Inhalt.sh
Digitales Muster mit Verzeichnisprüfung	-s, -i, -e, -m, -c Optional: -h, -v, -p	VerzeichnisPruefungPDF_Inhalt.bat und VerzeichnisPruefungPDF_Inhalt.sh

3.3 Konfigurationsdatei

Neben den im Abschnitt (3.2) genannten Übertragungsparametern und mitgelieferten Startskripten können in der Konfigurationsdatei übergreifende Einstellungen vorgenommen werden.

In diesem Abschnitt folgt eine allgemeine Einführung in den Aufbau einer Konfigurationsdatei. In den jeweiligen Konfigurationsdateien selbst existiert zu jedem Konfigurations-Element eine spezielle Beschreibung.

Die Pfadangaben in der Konfigurationsdatei müssen eventuell dem jeweiligen Betriebssystem angepasst werden. Die Konfigurationsdateien im Lieferumfang sind so voreingestellt, dass keinerlei Anpassungen nötig sind. Als Trennzeichen für Verzeichnisse wird das Zeichen '/' verwendet. Diese Voreinstellung erlaubt die Nutzung gleicher Konfigurationsdateien auf verschiedenen Betriebssystemen (Windows, Unix, Linux, ...). Relative Pfadangaben werden als relativ zum Installationsverzeichnis betrachtet. Die Konfigurationsdatei wird in 3 Abschnitte eingeteilt.

3.3.1 Abschnitt: Allgemeiner Teil

Im allgemeinen Teil werden allgemeine Informationen zum Prüflauf eingestellt.

3.3.1.1 Installationsverzeichnis

Das Installationsverzeichnis wird im Element „**pruefpfad**“ festgelegt.

z.B.: <pruefpfad> ./</pruefpfad>

3.3.1.2 Prüfdatenverzeichnis

Das Verzeichnis mit Prüfdateien wird im Element „**pruefdaten**“ festgelegt.

z.B.: <pruefdaten>Daten/</pruefdaten>

3.3.1.3 Geprüft-OK-Verzeichnis

Bei eingeschalteter Option –m wird die geprüfte Datei mit Status 'ok' in dieses Verzeichnis kopiert. Die Pfadangabe erfolgt im Element „**okdaten**“.

z.B.: <okdaten>Geprueft/NichtAbgelehnt/</okdaten>

3.3.1.4 Ausschlussverzeichnis

Bei eingeschalteter Option –m wird die geprüfte Datei mit Status 'abgelehnt' bzw. 'abbruch' in dieses Verzeichnis verschoben. Die Pfadangabe erfolgt im Element „**fehlerdaten**“.

z.B.: <fehlerdaten>Geprueft/Abgelehnt/</fehlerdaten>

3.3.1.5 Protokolldatei

Bei jedem Prüflauf wird eine Protokolldatei geschrieben. Der Pfad der Protokolldatei kann in der Konfigurationsdatei im Element „**log_datei**“ angegeben werden. Bei einem eventuellen fehlerhaften Programmablauf befinden sich in dieser Datei Informationen zur genauen Fehlerursache.

z.B.: <log_datei>Listen/XPM_Logfile.log</log_datei>

3.3.1.6 Dateifilter

Die Menge der zu prüfenden Dateien kann über das Element „**datei_filter**“ eingegrenzt werden.

z.B.: <datei_filter>*.LDT</datei_filter>

3.3.1.7 PDF-Dateifilter

Die Menge der zu prüfenden PDF-Dateien kann über das Element „**pdf_datei_filter**“ eingegrenzt werden.

z.B.: <pdf_datei_filter>*.PDF</pdf_datei_filter>

Hinweis: Sollte in der Pfadangabe des Schemas das ‘%’-Zeichen verwendet werden, so muss dieses durch die Zeichenkette ‘%25’ ersetzt werden.

3.3.1.8 Warnungen

Diese Einstellung ermöglicht es, das Protokollieren der Warnungen ein- bzw. auszuschalten. Die Konfiguration wird im Element „**warnungen**“ festgelegt.

Wertebereich: „ja“, „nein“.

z.B.: <warnungen>ja</warnungen>

3.3.1.9 Begrenzung der Fehlermeldungen

Standardmäßig begrenzt das XPM_LDK die gemeldeten Fehler einer bestimmten Mel dungsnummer auf eine festgelegte maximale Anzahl. Die maximale Anzahl kann je nach Prüfprojekt variieren und beträgt in der Regel zwischen 30 und 50. Diese Einstellung dient der Übersichtlichkeit eines Fehlerprotokolls, um die Liste nicht mit systematischen Fehlern zu überladen. Die standardmäßige Begrenzung kann ein- und ausgeschaltet werden.

Wertebereich: „ja“, „nein“.

z.B.: <fehler_begrenzen>ja</fehler_begrenzen>

3.3.1.10 Dokumentation

Mit dieser Einstellung kann auf eine alternative Dokumentationsdatei verwiesen werden, welche in der GUI unter „Hilfe / Hilfe“ verlinkt ist.

z.B.: <dokumentation>Doku/
KBV_ITA_AHEX_Handbuch_Pruefmodul_LDK_KV.pdf</dokumentation>

3.3.2 Abschnitt: Eingabedateien

Im Abschnitt Eingabedateien wird der Pfad zur Steuertabelle festgelegt. Die Steuertabelle für das XPM_LDK, im Element „**kbv_tabelle**“ spezifiziert, muss in der Konfigurationsdatei angegeben werden. In der Regel ist an dieser Steuertabelle durch den Anwender kein Eingriff notwendig.

z.B.: <kbv_tabelle> Bin/ldk_tabelle.bin</kbv_tabelle>

3.3.3 Abschnitt: Ausgabedateien

Im Abschnitt Ausgabedateien werden die Pfade für die Ausgabelisten und Protokolle festgelegt.

Über das Attribut Format wird das Ausgabeformat festgelegt:

- CSV Kommaseparierte Ausgabe, über das Attribut 'Trennzeichen' lässt sich das Trennzeichen zwischen den Spalten festlegen. Standardmäßig ist hier das Komma voreingestellt.
- HTML HTML-Format
- JRPRINT Internes Ausgabeformat, kann vom Prüfmodul angezeigt und gedruckt werden
- PDF Portable Document Format
- PRINTER Direktausgabe auf den Drucker
- PRINTER_DIALOG Direktausgabe auf den Drucker mit Einstellungsfenster
- RTF Rich Text Format, formatiertes Textformat
- TEXT ASCII Text, über das Attribut 'Seitenbreite' lässt sich die Seitenbreite in Zeichen festlegen. Standardmäßig ist hier die Breite von 80 Zeichen voreingestellt. Ein ansprechendes Layout erreicht man, in dem die Breite auf den Wert 120 gesetzt wird.
- XLS Microsoft Excel-Format
- XML XML-Format

Tabelle 1: Ausgabeformate

z.B.: <FehlerListe Format="PDF">Listen/Protokoll.pdf</FehlerListe>

PDF Dateien können angezeigt und gedruckt werden.

Alles, was dazu benötigt wird, ist der Adobe Reader®, der kostenlos unter <https://get.adobe.com/de/reader/> heruntergeladen werden kann. XPM_LDK erzeugt PDF Dokumente, die vom Acrobat Reader® ab der Version 5.0 und höher angezeigt werden können.

Beim Verarbeiten von Massendaten (Servermodus, zip-Archive) werden die Ausgabedateien vom Prüfmodul eigenständig umbenannt. Der Name der Ausgabedatei setzt sich zusammen aus dem Namen der Prüfdatei und den vorgegebenen Dateinamen. Diese Vorgehensweise verhindert das Überschreiben bereits erzeugter Protokolle.

Bei Fehlerprotokollen fügt das XPM noch ein Präfix hinzu, das den Errorlevel der Prüfung kennzeichnet. Es werden folgende Präfixe verwendet:

Errorlevel	Präfix
0	Ok_
1	Warnung_

2	Fehler_
3	Abbruch_

Tabelle 2: ErrorLevel

So wird bei einer Prüfdatei mit dem Namen ‘Test.xml’ und dem Errorlevel 1 der Name „Warnung_Test.xml.Fehler.pdf“ für die Fehlerliste vergeben.

Zusätzlich wurde eine zweite flexiblere Umbenennungsmethode integriert.

Sobald im Namen einer Ausgabedatei die Variable \${DATEI_NAME} verwendet wird, ersetzt XPM diese Variable durch den Namen der Prüfdatei. Fehlt die Variable \${DATEI_NAME}, dann wird der Name der Prüfdatei als erstes im Namen der Ausgabedatei eingefügt.

Bei der Umbenennung von Fehlerprotokollen kann man mit der Variablen \${STATUS} den Fehlerstatus an einer beliebigen Stelle im Dateinamen platzieren. Fehlt die Variable \${STATUS}, dann wird der Fehlerstatus als erstes im Namen der Ausgabedatei eingefügt.

3.3.3.1 Statistikdatei

Im Abschnitt Ausgabedateien Element ‘**Fehlerstatistik**’ wird der Pfad für eine Statistikliste festgelegt.

z.B.:

```
<StatistikListe Format="PDF">./Listen/FehlerStatistik.pdf</StatistikListe>
```

In der Statistikliste wird für jede Prüfdatei, für die eine Meldung des Prüfmoduls erfolgte, eine Meldungsstatistik ausgegeben. Die mit dem Status ‘ok’ geprüften Dateien tauchen in der Statistikliste nicht auf.

Die auftretenden Meldungstexte können das Zeichen '%s' enthalten. Dies ist kein Programmfehler sondern nur ein Hinweis darauf, dass diese Meldung variable Inhalte enthält, die erst zur Laufzeit ermittelt werden und unterschiedliche Ausprägungen enthalten kann.

z.B.: Zu der GNR '%s' wurde keine Angabe im Feld '%s' gemacht.

3.3.4 Umgebungsvariablen

Jedes Element der Konfigurationsdatei darf Umgebungsvariablen enthalten.

Diese Umgebungsvariablen müssen der JavaVM jedoch über den Übergabeparameter –D übergeben werden. Nach dem Einlesen der Konfigurationsdatei werden die Umgebungsvariablen durch ihre Werte ersetzt. Findet das XPM_LDK eine Umgebungsvariable nicht, wird der Prüflauf abgebrochen.

Mit Hilfe von Umgebungsvariablen kann mehreren Benutzern eine separate Umgebung zur Verfügung gestellt werden, die auf eine einzige Installation zugreifen.

Beispiel:

In der Konfigurationsdatei wird der Prüfpfad folgendermaßen festgelegt:

```
<pruefpfad>%INSTALLATION% / %UMGEBUNG% </pruefpfad>
```

Die zwei Umgebungsvariablen INSTALLATION und UMGEBUNG müssen entweder in einer Batchdatei bzw. einem Shellskript:

```
set INSTALLATION=C:\Projekte\JavaPruefmodul\Test  
set UMGEBUNG=LDK.Praxis
```

oder in der aufrufenden Applikation entsprechend gesetzt werden.

Jetzt muss nur noch dafür gesorgt werden, dass die Umgebungsvariablen der JavaVM bekannt sind.

Über den folgenden Aufruf werden die Umgebungsvariablen unter gleichem Namen dem XPM_LDK bekannt gegeben.

```
java -DINSTALLATION=%INSTALLATION% -DUMGEBUNG=%UMGEBUNG% ...
```

3.4 Arbeiten im Kommandozeilenmodus

Das Prüfmodul kann als ein kommandozeilenorientiertes Programm gestartet werden. Der Lauf des Prüfmoduls wird mithilfe von Übertragungsparametern gesteuert. Es folgt eine Auflistung aller Übertragungsparameter.

3.4.1 Übertragungsparameter

Übertragungsparameter	Beschreibung
-c	Das Prüfmodul braucht für die Prüfung die Pfadangabe einer XML-Konfigurationsdatei. Hinter dieser Option muss die Pfadangabe stehen!
-e	Das Prüfmodul wird im Einzellaufmodus gestartet. XPM_LDK verarbeitet alle Dateien eines Eingangsverzeichnisses und beendet sich anschließend. Diese Option ist nur in Kombination mit Servermodus aufrufbar. Dieser Übertragungsparameter ist optional.
-f	Hinter dieser Option sollte die Pfadangabe einer Prüfdatei stehen, die vom Prüfmodul bearbeitet wird. Dieser Übertragungsparameter ist optional.
-h	Das Prüfmodul gibt einen Hilfetext aus und beendet sich anschließend.

-m	Das Prüfmodul verschiebt bereits geprüfte Dateien bzw. Zip-Archive in entsprechende Verzeichnisse, die in der Konfigurationsdatei eingestellt werden. Dieser Übergabeparameter ist optional.
-p	Alle Konfigurationsmöglichkeiten der Konfigurationsdatei, können nun über diesen Übergabeparameter gesetzt werden. Die Zuweisung erfolgt in der Form 'Schalter[@Attribut]=Wert'. Schalter ist ein beliebiger Schalter der Konfigurationsdatei. Soll nur das Attribut eines Schalters gesetzt werden so muss der Name des Attributes hinter dem @-Zeichen angegeben werden. Der Wert selbst wird hinter dem Gleichheitszeichen angegeben. Dieser Parameter kann mehrfach übergeben werden, um diverse Einstellungen vorzunehmen. Bitte beachten Sie, dass die hier übergebenen Parameter die Einstellungen der Konfigurationsdatei überschreiben. Schauen Sie sich zum besseren Verständnis das Beispiel 4 an. Dieser Übergabeparameter ist optional.
-s	Das Prüfmodul wird im sogenannten Servermodus gestartet. XPM_LDK verarbeitet im 30 Sekunden-Takt Dateien eines Eingangsverzeichnisses. Der Abbruch des Programmlaufs kann über CTRL-C erfolgen. Dieser Übergabeparameter ist optional.
-v	Das Prüfmodul gibt die Versionsnummer des XPM-Kernels aus und beendet sich anschließend. Wird außer diesem Übergabeparameter auch noch die Konfigurationsdatei (Übergabeparameter -c) angegeben, so wird zusätzlich die Versionsnummer des Prüfpaketes ausgegeben.
-z	Hinter dieser Option sollte die Pfadangabe einer Zip-Datei stehen, die vom Prüfmodul bearbeitet wird. Der Inhalt des Zip-Archivs darf nur Prüfdateien enthalten. Dieser Übergabeparameter ist optional.
-i	Der Parameter -i aktiviert die inhaltliche Prüfung der PDF-Datei. Über den Pfad hinter dem Parameter -f wird angegeben welche Datei geprüft werden soll. Bsp: -i -f ./DigitaleMuster/xyz.pdf

Tabelle 3: Übergabeparameter

3.4.2 Beispiele

Nun folgen Beispiele für den Aufruf des XPM_LDK.

3.2.2.1. Beispiel 1: Prüfen einer einzelnen Datei

Übergabeparameter:

```
-c Konfig\konfig.xml -f Daten\Muster_10.pdf -i
```

XPM_LDK liest die Konfigurationsdatei 'Konfig/konfig.xml' ein und prüft die Datei 'Daten/Muster_10.pdf'. Anschließend wird XPM_LDK beendet.

3.2.2.2. Beispiel 2: Prüfen eines Verzeichnisses

Übergabeparameter:

```
-s -e -m -i -c Konfig\konfig.xml
```

XPM_LDK liest die Konfigurationsdatei 'Konfig/konfig.xml' ein und prüft das komplette Verzeichnis, welches in der Konfigurationsdatei unter dem Konfigurationsschalter 'pruefdaten' angegeben wurde. Anschließend wird XPM_LDK beendet.

3.2.2.3. Beispiel 3: Prüfen eines Verzeichnisses (ohne automatisches Beenden)

Übergabeparameter:

```
-c Konfig/konfig.xml -s -i
```

XPM_LDK liest die Konfigurationsdatei 'Konfig/konfig.xml' ein und prüft das komplette Verzeichnis, welches in der Konfigurationsdatei unter dem Konfigurationsschalter 'pruefdaten' angegeben wurde. Alle 30 Sekunden wird das angegebene Verzeichnis abgeprüft.

3.2.2.4. Beispiel 4: Setzen des Pfades und des Formates einer Ausgabeliste

Übergabeparameter:

```
-p FehlerListe=Listen/Protokoll.xml -p FehlerListe@Format=XML
```

Die Ausgabeliste mit dem Namen 'FehlerListe' bekommt den Pfad 'Listen/Protokoll.xml' zugewiesen. Das Format der Ausgabeliste wird auf PDF gesetzt.

4. Hinweise zu Java

4.1 Starten eines Java-Programms

Das LDK-Prüfmodul ist eine Java-Applikation und wird in einer Java Laufzeitumgebung ausgeführt.

Hier ein Beispiel für einen Aufruf:

```
java -Xmx300m
      -Dfile.encoding=8859_1
      -cp "Bin/*"
      de.kbv.pruefmodul.modul.ldk.start.StartKonsole
      -c Konfig/konfig.xml -f Daten/Muster_10.pdf
```

Der Befehl 'java' startet die virtuelle Maschine von Java.

Der Parameter '-Xmx300m' erlaubt der Java Laufzeitumgebung einen Hauptspeicher von bis zu 300 MB zu reservieren. Diese Option garantiert einen stabilen Programmablauf bei Abrechnungsdateien in der Größenordnung bis ca. 300 MB.

Der Parameter '-Dfile.encoding=8859_15' stellt den entsprechenden Zeichensatz ein und ermöglicht hier die Verwendung von deutschen Umlauten.

Der Parameter '-cp "Bin/*"' spezifiziert alle Java-Archive, die für den Programmablauf benötigt werden.

Der Parameter 'de.kbv.pruefmodul.modul.ldk.start.StartKonsole' ist der Name einer Klasse, die das XPM startet.

Die Parameter '-c' und '-f' sind die eigentlichen Übergabeparameter, die an das Prüfprogramm übergeben werden.

4.2 Starten des Prüfmoduls aus einem Java-Programm

Das LDK-Prüfmodul ist eine Java-Applikation und kann von einem anderen Java-Programm aufgerufen werden. Für eine leichtere Anbindung wurde die Klasse `de.kbv.pruefmodul.core.extern.XPMEinstieg` implementiert. Diese Klasse ist im Java-Archiv `xpm-core-<Versionsnummer>.jar` im Quellcode enthalten. Bitte schauen Sie sich hierzu die Methode 'main' etwas genauer an.

Beispielaufruf:

```
package de.org.beispiel

import de.kbv.pruefmodul.core.extern.XPMEinstieg;
import de.kbv.pruefmodul.modul.ldk.XPMAdapter;

public class XpmEinstiegTest {

    public static void main(String[] args) throws XPMException {
        // TODO Auto-generated method stub
        System.out.println("user.dir: " + System.getProperty("user.dir"));
        XPMEinstieg xpm = new XPMEinstieg("src/test/resources/Konfig/konfig.xml",
                                           "src/test/resources/Daten/Z01Auftrag.ldt", new XPMAdapter());

        xpm.setServer(false);
        xpm.setZipFile(false);
        int nStatus = xpm.pruefe();
        System.out.println("Einzel-Prüfung mit Status " + nStatus + " beendet.");
        System.out.println();

    }
}
```

5. Prüfregeln

In diesem Abschnitt werden die Prüfregeln aufgeführt, die das Prüfmodul bei der Analyse eines digitalen Musters prüft.

5.1 Allgemeine Prüfung bei Verwendung des Modus „Digitales Muster“ oder „Digitales Muster mit Verzeichnisprüfung“

Das Prüfmodul liest die Formularnummer aus den Metadaten der zu prüfenden PDF-Datei aus. Wird eine Formularnummer ausgelesen, die nicht „10“ oder „10a“ entspricht, gibt das Prüfmodul dies als Fehlermeldung zurück.

Liegt ein Muster 10 oder 10A vor, führt das Prüfmodul nachfolgend genannte Prüffälle aus. Ziel dieser Prüfungen ist es, die formale Korrektheit des jeweiligen Musters festzustellen.

Hinweis:

Beim Auslesen der Formularnummer des digitalen Musters aus den Metadaten unterstützt das Prüfmodul sowohl die Darstellung der Metadaten als eigene Elemente als auch als Attribute (Details siehe Kapitel 7.9.2.2 der XPM-Spezifikation). Somit werden die folgenden beiden Darstellungen unterstützt:

Darstellung mit Elementen:

```
<ftx:ControlData rdf:parseType="Resource">
  <control:Anzahl_Zeichen_Titel>0</control:Anzahl_Zeichen_Titel>
  <control:Anzahl_Zeichen_Vorname>0</control:Anzahl_Zeichen_Vorname>
  <control:Anzahl_Zeichen_Namenszusatz>0</control:Anzahl_Zeichen_Namenszusatz>
  <control:Anzahl_Zeichen_Hausnummer>0</control:Anzahl_Zeichen_Hausnummer>
  <control:Anzahl_Zeichen_Postleitzahl>0</control:Anzahl_Zeichen_Postleitzahl>
  <control:Anzahl_Zeichen_Wohnsitzlaendercode>0</control:Anzahl_Zeichen_Wohnsitzlaendercode>
  <control:Auftragsnummer_Einsender>0</control:Auftragsnummer_Einsender>
  <control:Formularnummer>10</control:Formularnummer>
  <control:Formularversion>07.2017</control:Formularversion>
</ftx:ControlData>
```

Darstellung mit Attributen:

```
<ftx:ControlData
  control:Anzahl_Zeichen_Titel="8"
  control:Anzahl_Zeichen_Vorname="14"
  control:Anzahl_Zeichen_Namenszusatz="13"
  control:Anzahl_Zeichen_Hausnummer="3"
  control:Anzahl_Zeichen_Postleitzahl="5"
  control:Anzahl_Zeichen_Wohnsitzlaendercode="1"
  control:Auftragsnummer_Einsender="0"
  control:Formularnummer="10"
  control:Formularversion="07.2017" />
```

5.1.1 Prüffälle für Muster 10

Liegt ein Muster 10 vor, führt das Prüfmodul nachfolgend genannte Prüffälle aus.

5.1.1.1 Aufbau der Kostenträgerkennung

Das Prüfmodul prüft, ob die in PDF-Feld „4111_Kostentraegerkennung“ angegebene Kostenträgerkennung genau neunstellig ist und ausschließlich aus Ziffern besteht.

Ist dies nicht der Fall, wird ein entsprechender Hinweistext ausgegeben.

5.1.1.2 BSNR-Prüfung

Das Prüfmodul prüft, ob die im PDF-Feld „0000_Betriebsstättennummer“ angegebene BSNR genau neun Stellen hat.

Im ASV-Fall (PDF-Formularfeld „0000_weitere Kennzeichen“ hat den Wert 1) prüft das Prüfmodul weiterhin ob die BSNR den folgenden Aufbau hat: 00nnnnnnP

Dabei ist „00“ ein fest vorgegebener Wert, „n“ jeweils eine Ziffer und P die Prüfziffer

Im TSS-Fall (PDF-Formularfeld „0000_weitere Kennzeichen“ hat den Wert 7) prüft das Prüfmodul weiterhin ob die BSNR den folgenden Aufbau hat: 35kknnnn

Dabei ist „35“ ein fest vorgegebener Wert, „n“ jeweils eine Ziffer und „kk“ ein Wert aus folgender Menge: 01-03, 06-21, 24, 25, 27, 28, 31, 37-73, 78-81, 83, 85-88, 93-96, 98, 99

Im SAPV - Fall (BSNR beginnt mit „74“) prüft das Prüfmodul weiterhin ob die BSNR den folgenden Aufbau hat: 74kknnnn63

Dabei sind „74“ und „63“ feste Werte, „n“ jeweils eine Ziffer und „kk“ ein Wert aus folgender Menge: 01-03, 06-21, 24, 25, 27, 28, 31, 37-73, 78-81, 83, 85-88, 93-96, 98, 99

In allen anderen Fällen prüft das Prüfmodul, ob die BSNR folgenden Aufbau hat: kknnnnnnmm

Dabei ist „n“ jeweils eine Ziffer „mm“ beliebig und „kk“ ein Wert aus folgender Menge: 01-03, 06-21, 24, 25, 27, 28, 31, 37-73, 78-81, 83, 85-88, 93-96, 98, 99

Bei einer fehlerhaften Prüfung wird ein entsprechender Hinweis ausgegeben.

5.1.1.3 BSNR-Erstveranlasser-Prüfung

Das Prüfmodul führt die Prüfungen durch, die schon im Abschnitt 5.1.1.2 beschrieben wurden. Die Unterscheidung der Fälle erfolgt jeweils durch die ersten beiden Ziffern der BSNR:

- 00 entspricht ASV-Fall
- 35 entspricht TSS-Fall
- 74 entspricht SAPV-Fall
- Alle anderen Werte sind alle hier nicht erwähnten Fälle

Ist das nicht der Fall wird ein entsprechender Hinweis ausgegeben.

5.1.1.4 LANR-Prüfung

Das Prüfmodul prüft ob die im PDF-Feld „0000_LebenslangeArztnummer“ eingetragene LANR genau neunstellig ist.

Im ASV-Fall (PDF-Formularfeld „0000_weitere Kennzeichen“ hat den Wert 1) prüft das Prüfmodul, ob die LANR folgenden Aufbau hat: 555555nff

Dabei ist „n“ eine Ziffer, „ff“ der Fachgruppencode und „555555“ ein fester Wert.

Im Nicht-ASV-Fall prüft das Prüfmodul darüber hinaus, ob die LANR folgenden Aufbau hat: nnnnnnmff

Dabei ist „n“ jeweils eine Ziffer, „m“ die Prüfziffer und ff der nach Anlage 35 des BAR-Schlüsselverzeichnisses erlaubte Inhalt mit „00“ als Ersatzwert.

Ist das nicht der Fall wird ein entsprechender Hinweis ausgegeben.

5.1.1.5 LANR-Erstveranlasser-Prüfung

Das Prüfmodul prüft das PDF-Formularfeld „0000_Erstveranlasser_LANR“ gemäß der im Abschnitt 5.1.1.4 beschriebenen Prüfung. In diesem Fall ergibt sich der ASV-Fall, wenn die angegebene LANR mit „555555“ beginnt.

Ist das nicht der Fall wird ein entsprechender Hinweis ausgegeben.

5.1.1.6 Aufbau der Versichertennummer

Das Prüfmodul prüft die Versichertennummer im PDF-Feld „0000_Vericherten_ID“ daraufhin, ob diese mit einem Buchstaben beginnt und anschließend neun Ziffern folgen.

Ist das nicht der Fall wird ein entsprechender Hinweis ausgegeben.

5.1.1.7 Wertebereich des Statusfelds

Das Prüfmodul prüft, ob Werte für die Eintragungen nur die erlaubten Wertebereiche einhalten.

- Versichertenart (PDF: 3108_Versichertenart): 1, 3, oder 5

- Besondere Personengruppe (PDF: 4131_BesonderePersonengruppe): „leer“, 4, 6, 7, 8, 9 oder „leer“
- DMP-Kennzeichnung (PDF: 4132_DMP_Kennzeichnung): „leer“, 1, 2, 3, 4, 5, 6 ^
- ASV_TSS_Kennzeichen: (PDF: 000_weitere_Kennzeichen): „leer“, 1, 7 oder „leer“

Ist das nicht der Fall gibt das Prüfmodul jeweils einen entsprechenden Hinweistext aus.

5.1.1.8 Signaturprüfung

Das Prüfmodul prüft, ob im Feld „0000_QES“ eine elektronische Signatur vorhanden ist. Dazu prüft die Prüfsoftware, ob für das PDF-Feld „0000_QES“ ein Signature Dictionary als /v-Wert vorhanden ist.

Sollte das Signature Dictionary nicht vorhanden sein, wird die Fehlermeldung ausgegeben: „Das PDF-Dokument enthält keine QES im vordefinierten QES-Feld“.

Des Weiteren prüft das Prüfmodul, ob die QES hinsichtlich der Integrität des Dokumentes unverändert ist. Sollte die Prüfung fehlerhaft sein, wird die Meldung ausgegeben: „Das Dokument wurde nach dem Aufbringen der elektronischen Signatur geändert. Die Integrität des Dokumentes wurde nach dem Aufbringen der Signatur gebrochen.“

5.1.2 Prüffälle für Muster 10A

Liegt ein Muster 10A vor, führt das Prüfmodul nachfolgend genannte Prüffälle aus.

Für ein Muster 10A führt das Prüfmodul folgende Prüfungen durch, die bereits für Muster 10 beschrieben wurden:

- Aufbau der Kostenträgerkennung (Abschnitt: 5.1.1.1)
- BSNR-Prüfung (Abschnitt: 5.1.1.2)
- LANR-Prüfung (Abschnitt: 5.1.1.4)
- Aufbau der Versichertennummer (Abschnitt: 5.1.1.6)
- Wertebereich des Statusfelds (Abschnitt: 5.1.1.7)

5.1.2.1 Sonstige Aufträge gefüllt

Das Prüfmodul prüft, ob das PDF-Feld „4205_Auftrag61“ angekreuzt ist. Ist dies der Fall darf das PDF-Feld „4205_Auftrag61_sonstige_Auftraege“ nicht leer sein.

Ist das Feld leer, wird eine entsprechende Hinweismeldung ausgegeben.