PJ-Mapper fachliche und technische Übersicht

Version 0.1 | Stand **05.04.2024**

GEMEINSAM. DIGITAL. VERNETZT.

Dokumenteninformation	
Programm	Polizei 20/20
Programmleiter	Holger Gadorosi
Projektleiter/Verantwortlicher	Dr. Michael Mazur
Dokumententitel	PJ-Mapper Kurzinformation
Version	0.1
Erstellt am **Erstellt von**	05.04.2024 Babette Borries
Zuletzt bearbeitet am **Zuletzt bearbeitet von**	<TT.MM.JJJJ> <Vorname Nachname (OE)>
Status	In Bearbeitung

Vorprüfung des Dokumentes		
Fachliche Prüfung (Fachlichkeit)	Wählen Sie ein Element aus.	am <TT.MM.JJJJ>
Technische Prüfung (Technik)	Wählen Sie ein Element aus.	am <TT.MM.JJJJ>
Methodische Prüfung (PMO)	Wählen Sie ein Element aus.	am <TT.MM.JJJJ>
<Bitte tragen Sie hier weitere mitzeichnungspflichtige Prüfinstanzen je nach Thematik und festgelegtem Erstellungsprozess ein, z.B. AG Recht, AG Vergabe.>		am <TT.MM.JJJJ>

Freigabe Hauptversion 1.0 durch Freigabeverantwortlichen bzw. zuständiges Gremium
Freigegeben am: <TT.MM.JJJJ>
Freigegeben durch: <Gremiensitzung, TOP Nr.>

Versionshistorie			
Version	**Datum**	**Erstellt durch**	**Inhaltliche Kurzbeschreibung der Neuerungen**
0.1	**Erstellt am** **Erstellt von**	05.04.2024 Babette Borries	

Referenzierte Dokumente			
Referenzdokument	**Version**	**Autor**	**Datum**
<A>	<Link>	<Vorname Nachname,OE >	<TT.MM.JJJJ>

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Bisherige Kommunikations- und Bearbeitungsprozesse in Strafsachen zwischen Polizei und Justiz beruhen primär auf dem Austausch von Papierakten. Das Ziel des Projekts „digitaler Austausch Polizei und Justiz“ (dAPJ) ist daher die Entwicklung einer Schnittstelle für die Etablierung einer medien-bruchfreien digitalen Kommunikation zwischen Polizei und Justiz in Strafsachen in einem gemeinsamen Projekt mit der Justiz als Teil des Programms P20.

Das Projekt dAPJ des Bundeskriminalamts (BKA) bildet mit der Entwicklung, Testung und Bereitstellung des „PJ-Mappers“ einen zentralen Bestandteil der zu etablierenden Kommunikationskette zwischen Polizei und Justiz. Dieser Mapper übersetzt die Standards XPolizei und XJustiz bidirektional und stellt so die Grundlage für die digitale Kommunikation in Strafsachen dar.

Dazu bietet der PJ-Mapper eine REST-API für die Übermittlung und den Empfang von XML-Dokumenten an, die von einem Quell- in ein abweichendes Zielformat konvertiert werden. Dabei wird vorausgesetzt, dass die Struktur der vom Mapper zu verarbeitenden XML-Dokumente als XML Schema Definition (XSD) bekannt ist. Unterschiedliche XSD-Versionen eines Nachrichtentyps werden unterstützt.

Als Hilfsmittel für die Übertragung der Dokumenteninhalte von der Quell- in die Zieldatei werden Template-XML-Dateien (kurz *Templates*) genutzt, die mit Platzhaltern und sogenannten „Processing Instructions“ (kurz *PI*) angereichert sind. Zusätzlich werden die von der Polizei zur Verfügung gestellten *Codelisten* verwendet, um z.B. für Staaten, Sprachen oder Währungen die für das Zielsystem relevanten Codes zu ermitteln.

2. Übersicht Produktivsystem

In der produktiven Umgebung erfolgt die Kommunikation mit der REST-API des PJ-Mappers nicht direkt mit den Polizei- und Justiz-Systemen, sondern über eine externe Zwischeninstanz: eKoPol oder e²A.

Codelisten werden über einen externen *Katalogdienst* zur Verfügung gestellt. Aktualisierungen für Templates und XML-Schemata werden über einen *S3 Bucket* bereitgestellt.

Abbildung 1: Systemübersicht Produktivsystems

3. Externe Systeme

Der PJ-Mapper besitzt Schnittstellen zu externen Systemen, die in diesem Kapitel beschrieben werden.

3.1. eKoPol / e²A

Die Kommunikation mit der REST-API des PJ-Mappers erfolgt über eKoPol oder e²A. eKoPol bzw. e²A übermittelt per HTTP POST Request ein XPolizei- oder XJustiz-XML-Dokument, welches vom PJ-Mapper in das Format des anderen Systems, also XJustiz- oder XPolizei, konvertiert werden soll.

Der PJ-Mapper gibt folgende HTTP Response Codes zurück:

- 200 (OK): das Mapping war erfolgreich. Der HTTP Message Body enthält das gemappte XML-Dokument.
- 422 (Unprocessable Content): das Mapping konnte nicht durchgeführt werden. Der HTTP Message Body enthält als plain/text Informationen zur Fehlerursache.

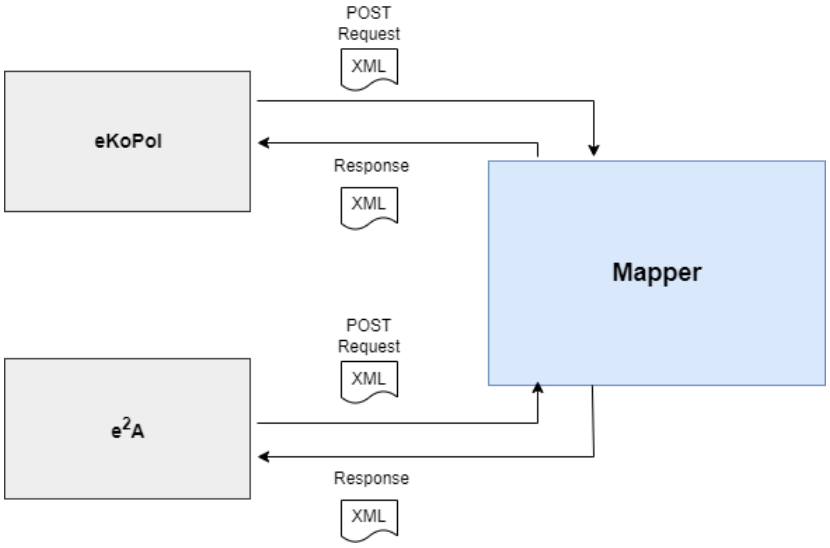

Abbildung 2: Schnittstelle zu eKoPol / e2A

3.2. Katalogdienst

Sowohl XPolizei als auch XJustiz verwenden in XML-Nachrichten codierte Informationen, z.B. für Staaten, Sprachen oder Währungen. Codelisten werden für die Übersetzung der Codes von einem in das andere System genutzt. Sie werden über einen externen *Katalogdienst* zur Verfügung gestellt.

Der PJ-Mapper fragt per HTTP GET Request den externen Katalogdienst gezielt nach Code Mappings. Diese werden im json-Format in der HTTP-Response zurückgegeben.

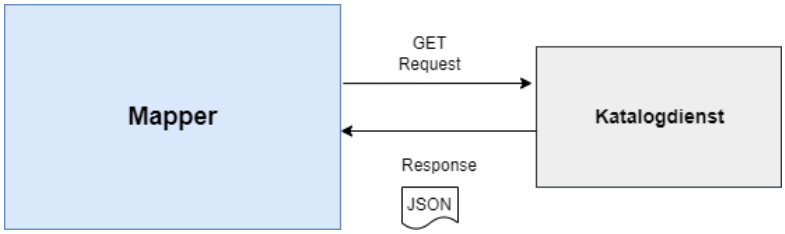

Abbildung 3: Schnittstelle zum Katalogdienst

3.3. S3 Bucket

Aktualisierungen für Templates, XML-Schemata und das Codelistenverzeichnis werden dem PJ-Mapper in einem jar-File über einen (*MinIO) S3 Bucket* (Objektspeicherdienst) bereitgestellt.

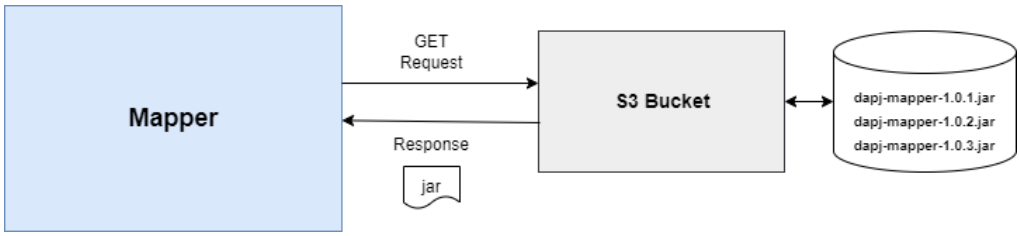

Abbildung 4: Schnittstelle zu S3 Bucket

4. PJ-Mapper

Der dynamische / generische PJ-Mapper hat folgende Ziele:

1) Einfach zu lesende, anzupassende und möglichst XML nahe Darstellung der Mappings (hier *Templates* genannt).
2) Parallele Unterstützung mehrere Versionen eines Templates. Ein Template repräsentiert ein Mapping zwischen zwei Teildatenmodellen. Die Versionen der Templates können als eine Kombination zweier Versionen der XML-Schemas (Input und Output) angesehen werden.
3) Die Abbildung der fachlichen Mappings wird vollständig getrennt von dem technischen PJ-Mapper gehalten. Damit kann ein fachliches Update „on the fly“ vollzogen werden, ohne dass die Instanzen des PJ-Mappers heruntergefahren, ausgetauscht und wieder hochgefahren werden müssen. Des weiteren ist es sehr einfach, ein „Rollback“ zu einer früheren Version durchzuführen, falls das neuere Update doch zurückgenommen werden muss.

4.1. Arbeitsweise des PJ-Mappers

Der PJ-Mapper parst zunächst das Quell-XML-Dokument und lädt es als DOM-Baumstruktur („Quell-DOM-Objekt“) in den Speicher. Aus den Angaben der Quelle wird versucht, Quell- und Ziel-XML-Version zu bestimmen. Falls dies nicht gelingt, werden Default-Versionen verwendet. Auch das XML-Schema der Quelle wird ermittelt.

Mit diesen Informationen wird ein passendes *Template* (siehe nächstes Kapitel) für die Generierung der Ausgabe bestimmt und als „Gerüst“ des „Ziel-DOM-Objekts“ in den Speicher geladen, sowie das XML-

Schema des Ziels ermittelt. Weiterhin wird ein sogenannter *Mapping Context* erzeugt und mit Informationen für das nachfolgende Mapping gefüllt.

Zu Durchführung des Mappings wird zuerst das Quell-DOM-Objekt mit dem zuvor ermittelten XML-Schema validiert. Nur ein valides Quell-Dokument kann weiterverarbeitet werden.

Das eigentliche Mapping wird mit dem *Generator* durchgeführt. Dafür geht dieser das ermittelte Template durch und führt die im Template angegebenen Anweisungen aus.

Danach wird das Ziel-DOM-Objekt mit dem zuvor bestimmten XML-Schema validiert.

Im Erfolgsfall wird das Ziel-DOM-Objekt als XML-Dokument serialisiert und als Ergebnis des Mappings zurückgegeben.

4.2. Systemkomponenten

Der PJ-Mapper besteht aus mehreren Komponenten:

- **dapj-config**: fachspezifische XPolizei / XJustiz Templates, XML Schema Definitionen und Implementierungen zu Mapper-spi Funktionen; Aktualisierung via S3 Bucket
- **mapper-spi:** Interfaces und abstrakte Java Klassen
- **mapper**: PJ-Mapper Kernfunktionen, unabhängig von der Fachlichkeit, inklusive REST-API, die von eKoPol und e²A genutzt wird, sowie Kommunikation mit dem externen Katalogdienst zum Ermitteln und Aktualisieren von Codelisten

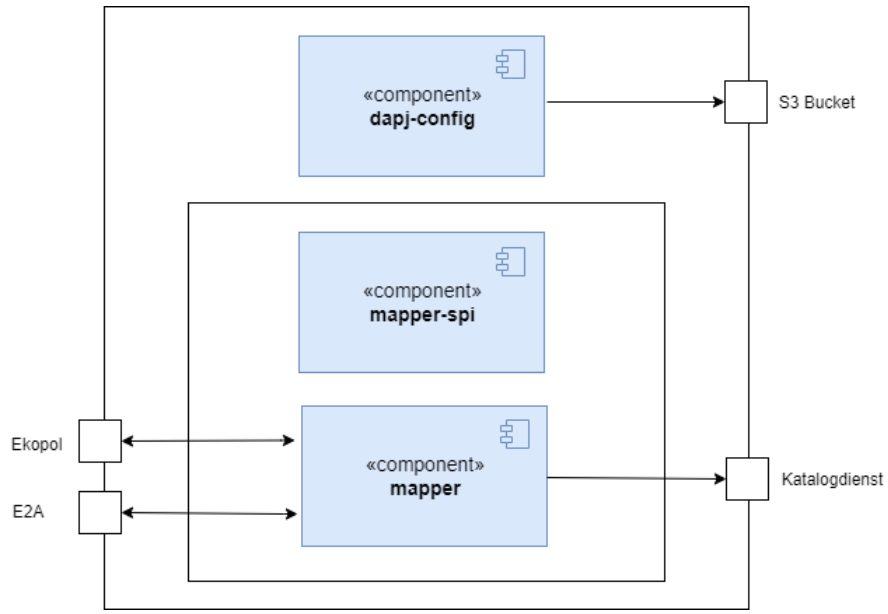

Abbildung 5: PJ-Mapper Komponenten

5. Templates

Als Hilfsmittel für die Übertragung ausgewählter Dokumentinhalte vom Quell- in das Ziel-XML-Dokument werden Template-XML-Dateien (kurz *Templates*) genutzt. Zusätzlich werden *Code Mappings* verwendet, um die für das Zielsystem relevanten Codes zu ermitteln.

Das Template ist ein XML-Dokument, welches das Ergebnis (das ausgehende XML-Dokument) beschreibt. Dieses enthält u.a. Elemente, Attribute oder Textinhalte, die erst über eine Anweisung gefüllt werden können. Weiterhin ist es manchmal notwendig, zwei oder mehr verschiedene Varianten des Inhalts eines Elements zu beschreiben, wobei am Ende nur eine davon in das ausgehende XML-Dokument generiert wird.

Damit der PJ-Mapper Daten aus dem Quelldokument in das Zieldokument übertragen oder Entscheidungen treffen kann, welche Teile des Templates ins Ergebnis kommen, muss das Template neben statischen Elementen auch dynamische Elemente (Anweisungen) enthalten. Diese Anweisungen (Kommandos) werden als XML „Processing Instructions“ definiert. XML Processing Instructions (PIs) werden in das Template eingefügt, um den PJ-Mapper dahingehend zu instruieren, wie XML-Elemente und -Attribute in das Ziel-XML-Dokument eingefügt werden sollen.

Zusätzlich können im Template Funktionen eingebunden werden, die z.B. dazu dienen, Daten zu konvertieren, Rechenoperationen auszuführen oder einzelne Element- und Attribut-Werte des Quell-DOM-Objekts zu lesen. Ebenso ist die Verarbeitung ganzer „Teil-Bäume“ des Quell-DOM möglich.