Spezifikation

JUSTIZ
JUSTIZRESSORT

Bundesministerium für Justiz

Elektronischer Rechtsverkehr (ERV)

EUM Validierungsmodul Spezifikation

Dateiname: EUM_Validierungsmodul_Spezifikation
Version: 1.5 vom 08.06.2017
Ersteller: Ronald Wischnack (ronald.wischnack@brz.gv.at)

1 Dokumentinformation

1.1 Inhaltsverzeichnis

1.2 Änderungsverlauf

Version	Datum	Ersteller	Kommentar
1.0	01.05.2014	Wischnack	Dokument erstellt
1.1	27.05.2014	Wischnack	Erweiterung Benutzte Bibliotheken
1.2	20.06.2014	Wischnack	Korrekturen Schemata, Erweiterte Beschreibung Schnittstelle
1.3	20.10.2014	Wischnack	Erweiterung um Schnittstelle ERV – Nachricht validieren
1.4	06.11.2014	Wischnack	Erweiterung Eingangsschnittstelle
1.5	08.06.2017	Wischnack	Anpassungen für Release 07.2017

2 Einleitung

2.1 Zweck des Dokuments

Dieses Dokument beschreibt die Funktionsweise des Validierungsmoduls für die EU-Mahnverfahren (folglich kurz EUM) Applikation in der jeweils gültigen Variante. Es muss mit der Veröffentlichung des EU-Mahnverfahrens mitveröffentlicht werden.

3 Guidelines zu Fehlermeldungen

1. Die Id einer Fehlermeldung der EUM-Validierung hat folgendes Format: **EUPO** gefolgt von einem Bindestrich ➖ und der **dreistelligen F ehlernummer** (mit führenden Nullen). Diese Nummer ist grundsätzlich ident zur Nummer der jeweiligen Geschäftsregel der EUM-Validierung (Beispiel: EUPO-001).
2. Zu jeder Fehlermeldung gibt es eine Beschreibung der Fehlerursache. Variablen im Fehlertext sind mit {999} gekennzeichnet.
3. Die Validierung basiert auf dem Veröffentlichungsdokument **EUM_Spezifikation_V3.1.5.pdf**. Alle dort erfassten Fehlermeldungen und Prüfungen wurden auch in diesem Modul erfasst. Abweichungen werden hier mit Grund dokumentiert.

4 Technische Anwendung der Bibliothek

4.1 Validierung des Payloads

Die Überprüfungen werden als JAR Bibliothek zur Verfügung gestellt und können über die Klasse

```
at.gv.brz.sta.validator.impl.EUPOMessageValidatorImpl
```

aufgerufen werden.
Als Übergabe wird, analog dem Inhalt der ERV-Nachricht, ein IncomingPayload erwartet.
Diese führt alle zutreffenden Prüfungen aus und liefert ein ValidationResult zurück. Dieses beinhaltet immer genau einen Eintrag.

1. Im Gutfall (kein Fehler): EUPO000 – Der elektronische Datensatz wurde angenommen
2. Im Fehlerfall: Code – Fehlermeldung

4.1.1 Aufrufbeispiel

```
EUPOMessageValidatorImpl impl = new EUPOMessageValidatorImpl();
ValidationResult result = impl.validateEupoMessage(byteArrayMsgUTF8);
```

4.1.2 Signatur

die Aufzurufende Methode enthält folgende Signatur:

```
ValidationResult validateEupoMessage(final byte[] message)
```

Als Ergebnis wird immer ein ValidationResult zurückgegeben. Dieses Beinhaltet IMMER genau einen Inhalt (EUPO000 – Der elektronische Datensatz wurde angenommen.).
Die einzelnen Prüfungen sind im Paket at.gv.brz.sta.validator.rules erfasst und können auch einzeln aufgerufen werden.

4.2 Validierung der ERV – Nachricht als SOAP - Nachricht - Format ERV 2.3

Diese neue Variante erlaubt es, die ERV Nachricht für das EUM vor Versendung komplett validieren zu lassen. Der Aufruf und die Rückmeldung sind ähnlich der Validierung des Payloads, nur dass in diesem Fall die gesamte Nachricht inklusive Anhängen und Metadaten geprüft wird.
Die Überprüfungen werden als JAR Bibliothek zur Verfügung gestellt und können über die Klasse

```
at.gv.brz.sta.validator.impl.EUPOMessageValidatorImpl
```

aufgerufen werden. Als Übergabe wird eine ERV-Nachricht in Format ERV 2.3 erwartet.
Diese führt alle zutreffenden Prüfungen aus und liefert ein ValidationResult zurück. Dieses beinhaltet immer genau einen Eintrag.

1. Im Gutfall (kein Fehler): EUPO000 – Der elektronische Datensatz wurde angenommen
2. Im Fehlerfall: Code – Fehlermeldung

4.2.1 Aufrufbeispiel

```
SOAPMessage soapMessage = …….
EUPOMessageValidatorImpl impl = new EUPOMessageValidatorImpl();
ValidationResult result = impl.validateERVEupoMessage(soapMessage);
```

4.3 Validierung der ERV - Nachricht im ERV 3.0 Format

Diese neue Variante erlaubt es, die ERV Nachricht für das EUM vor Versendung komplett validieren zu lassen. Der Aufruf und die Rückmeldung sind ähnlich der Validierung des Payloads, nur dass in diesem Fall die gesamte Nachricht inklusive Anhängen und Metadaten geprüft wird.
Die Überprüfungen werden als JAR Bibliothek zur Verfügung gestellt und können über die Klasse

```
at.gv.brz.sta.validator.impl.EUPOMessageValidatorImpl
```

aufgerufen werden. Als Übergabe wird eine ERV-Nachricht in Format ERV 3.0 erwartet.
Diese führt alle zutreffenden Prüfungen aus und liefert ein ValidationResult zurück. Dieses beinhaltet immer genau einen Eintrag.

1. Im Gutfall (kein Fehler): EUPO000 – Der elektronische Datensatz wurde angenommen
2. Im Fehlerfall: Code – Fehlermeldung

4.3.1 Aufrufbeispiel

1) Als ERV 3.0 Nachricht:

```
at.gv.justiz.erv.ervnachrichtextern.v3_0.ERVNachrichtTyp message = ....
EUPOMessageValidatorImpl impl = new EUPOMessageValidatorImpl();
ValidationResult result = impl.validatePreparedERV30Message(message);
```

2) Als byteArray

```
byte[] message = .... //(im Format
at.gv.justiz.erv.ervnachrichtextern.v3_0.ERVNachrichtTyp)
EUPOMessageValidatorImpl impl = new EUPOMessageValidatorImpl();
ValidationResult result = impl.validatePreparedERV30Message(message);
```

4.4 Validierung mit interner Datenstruktur

Wenn es aus diversen Gründen nicht, oder nur mit sehr hohem Aufwand möglich sein sollte, eine aufbereitete SOAP – Nachricht in die Validierung zu senden, bietet das Validierungsmodul ab der Version 1.1.11 die Möglichkeit, die Validierung mit einer EUM – eigenen Internen Datenstruktur aufzurufen. Das Ergebnis ist analog aller anderen Aufrufe.

4.4.1 Übergabestrukturen:

```
at.gv.brz.sta.validator.util.ERVMessage
 private byte[] messagePayload;
 private LinkedList<AttachmentTuple> attachmentTuples;
```

Dieses ist das bereitgestellte Trägerobjekt es enthält 2 Member: Den Payload als byte[] und ein weiteres internes Objekt als Liste, in welchem die Anhänge und Metadaten enthalten sind.

```
at.gv.brz.sta.validator.util.AttachmentTuple
 private byte[] attachment;
 private EUMAnhangInfo anhanginfo;
```

Dieses Trägerobjekt besteht aus 2 Teilen: Dem eigentlichen Anhang als byte[] und der dazugehörigen Anhanginfo als EUMAnhangInfo Objekt.

4.4.2 Aufrufbeispiel

```
EUMAnhangInfo metadata ...
 byte[] byteArraymessagePayload ...
 byte[] byteArrayAttachment ...
 ERVMessage message = new ERVMessage();
 message.setMessagePayload(byteArraymessagePayload);
 AttachmentTuple tuple = new AttachmentTuple();
 tuple.setAttachment(byteArrayAttachment);
 tuple.setAnhanginfo(metadata);
 message.addAttachment(tuple);
 EUPOMessageValidatorImpl impl = new EUPOMessageValidatorImpl();
 ValidationResult result = impl.validatePreparedERVEupoMessage(message);
```

4.4.3 Signatur

```
ValidationResult validatePreparedERVEupoMessage(ERVMessage message);
```

4.5 Validierung mit der Struktur aus den erv_commons (analog VJ-Validierung)

Dieser Teil der Implementierung wurde entfernt.

4.6 Externe Bibliotheken

Kurzname	Version	Offiziell zu finden unter
Slf4j-api	1.6.6	http://www.slf4j.org/dist/
Slf4j-ext	1.6.6	http://www.slf4j.org/dist/
Slf4j-log4j12	1.6.6	http://www.slf4j.org/dist/
Log4j	1.2.17	http://logging.apache.org/log4j/1.2/download.html
xercesImpl	2.10.0	http://tweedo.com/mirror/apache//xerces/j/
Xml-apis	1.4.01	http://tweedo.com/mirror/apache//xerces/j/
Jms	1.1	Maven
~~Erv_commons~~	~~1.3.0~~	~~http://www.kundmachungen.justiz.gv.at/edikte/km/kmhlp05.nsf/all/erv!OpenDocument~~ ~~Unter Punkt: ERV Verfahrensautomation Justiz~~