

TD MOA-ZS

Technische Dokumentation zu MOA-ZS

Dokumentinformation

Bezeichnung	Technische Dokumentation zu MOA-ZS
Kurzbezeichnung	TD MOA-ZS
Version	1.2.0
Datum	09.02.2009
Dokumentenklasse	Technische Dokumentation
Dokumentenstadium	Interner Entwurf.
Kurzbeschreibung	Dieses Dokument beschreibt die technischen Details der Applikation MOA-ZS
Autoren	DI Arne Tauber Nikolaus Gradwohl
Arbeitsgruppe	MOA-ZS

4	Inhalt	
5	1. Allgemeines	4
6	1.1 Synchroner Teil	4
7	1.1.1 Übernahme von Zustellstücken	4
8	1.1.2 Vollständigkeitsprüfung.....	5
9	1.1.3 Umrechnung der bPK	5
10	1.1.4 Prüfen der Adressierbarkeit	5
11	1.2 Asynchroner Teil	5
12	1.2.1 Aufbereiten des Zustellstücks	7
13	1.2.2 Signieren des Zustellstücks	7
14	1.2.3 Erstellung eines S/MIME Containers	7
15	1.2.4 Übergabe an den Zustellserver.....	8
16	1.2.5 Benachrichtigung des Absenders	8
17	2. Detaillierte Beschreibung von MOA-ZS.....	9
18	2.1 Funktionen	9
19	2.1.1 Vollständigkeitsprüfung.....	9
20	2.1.2 Lastverteilung	9
21	2.1.3 Auswahl des Zustellservers	9
22	2.1.4 Watchdog-Thread.....	10
23	2.1.5 Signaturerstellung.....	10
24	2.1.6 Verschlüsselung	10
25	2.2 Schnittstellen.....	10
26	2.2.1 Applikation an MOA-ZS	10
27	2.2.2 MOA-ZS an Applikation	14
28	2.2.3 Zusekopf.....	14
29	2.2.4 Stammzahlenregister.....	14
30	2.2.5 MOA-SS	15
31	2.2.6 Zustellserver	15
32	2.3 Datenorganisation	15
33	2.3.1 Tabelle payload	Error! Bookmark not defined.
34	2.3.2 Tabelle xmldocument.....	Error! Bookmark not defined.
35	2.3.3 Tabelle rcpt.....	Error! Bookmark not defined.
36	2.3.4 Tabelle identification	Error! Bookmark not defined.
37	2.3.5 Tabelle internetaddress	Error! Bookmark not defined.
38	2.3.6 Tabelle telephone	Error! Bookmark not defined.
39	2.3.7 Tabelle postaladdress.....	Error! Bookmark not defined.
40	2.3.8 Tabelle Status	Error! Bookmark not defined.
41	2.3.9 Tabelle delivery_quality	Error! Bookmark not defined.
42	2.3.10 Tabelle zuseserver	Error! Bookmark not defined.

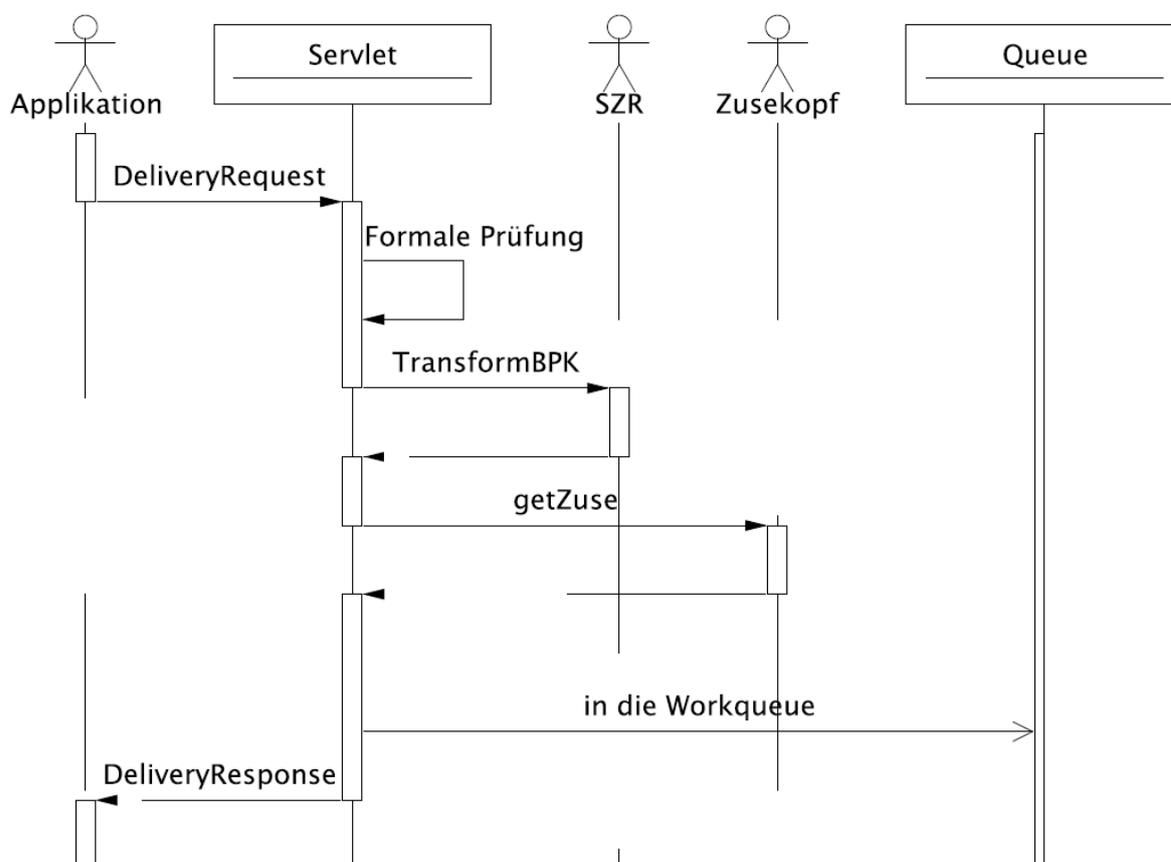
43	2.3.11	Tabelle queue_info	Error! Bookmark not defined.
44	3.	Referenzen	16
45	4.	Historie.....	17
46	5.	Anhang	18
47	5.1	Begriffe und Abkürzungen	18
48	5.2	Tabellenverzeichnis.....	18
49	5.3	Abbildungsverzeichnis.....	18
50	5.4	Konfigurationsdatei moazs_config.xml	19
51			

52 1. Allgemeines

53 MOA-ZS verarbeitet Zustellstücke in zwei Phasen. Zunächst werden die Zustellstücke in
54 einer synchronen Phase angenommen und überprüft, dann werden sie in einer asynchronen
55 Phase, ohne Verbindung zur Absenderapplikation aufbereitet und an den Zustellserver
56 übergeben.

57 1.1 Synchroner Teil

58 Abbildung 1, „Synchroner Teil der Requestverarbeitung“ zeigt den Ablauf der
59 synchronenPhase, der im Folgenden detailliert beschrieben wird.



60

61

Abbildung 1: Synchroner Teil der Requestverarbeitung

62

63 1.1.1 Übernahme von Zustellstücken

64 Zustellstücke werden von der Absenderapplikation an MOA-ZS mit einem SOAP-Request
65 übergeben. Dabei wird eine Nachricht im `DeliveryRequest` Format erstellt. XML-
66 Dokumente, die mit MOA-ZS zugestellt werden sollen, werden direkt in den Request als
67 Kindsknoten eingefügt. Sollen Binärdateien verschickt werden, gibt es drei Möglichkeiten der
68 Übergabe an MOA-ZS:

- 69 • Übergabe als `BinaryDocument` im Base64-Encoding
- 70 • Übergabe als SwA-Attachment
- 71 • Übergabe als Callback-URI

72 Eine genaue Beschreibung der Schnittstelle mit Beispielen befindet sich im Kapitel 2.2
73 Schnittstellen.

74 1.1.2 Vollständigkeitsprüfung

75 Ein Servlet übernimmt den SOAP-Request als `DOM` entgegen und überprüft die im Schema
76 bzw. in der Spezifikation geforderten Pflichtfelder. Wenn eine der Pflichtfeldprüfungen
77 fehlschlägt, wird eine Fehlermeldung generiert und an die Absenderapplikation
78 zurückgeschickt.

79 1.1.3 Umrechnung der bPK

80 Falls die Vollständigkeitsprüfung erfolgreich ist, wird versucht, einen Zustellserver für den
81 Empfänger zu finden. Dabei wird zuerst geprüft, ob eine bPK mitgeschickt worden ist. Falls
82 es sich nicht um eine (verschlüsselte) bPK des Bereiches "Zustellung" handelt, wird das
83 Stammzahlregister dazu verwendet, um mittels dessen Funktion `TransformBPK` die
84 übergebene bPK in eine verschlüsselte bPK des Bereichs „Zustellung“ umzurechnen.

85 1.1.4 Prüfen der Adressierbarkeit

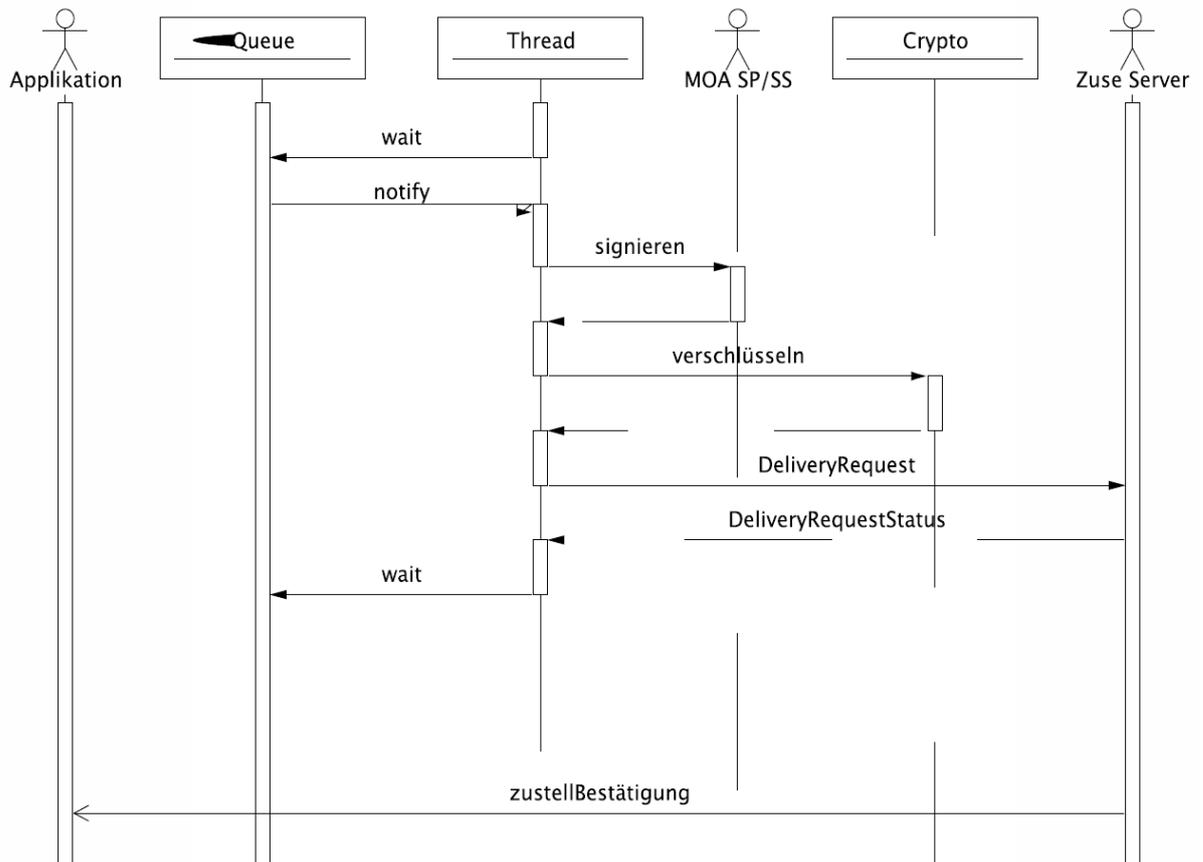
86 MOA-ZS stellt eine Anfrage an den Zustellkopf, ob der Empfänger elektronische
87 Zustellungen entgegennehmen kann und wählt aus der Liste der zurückgegebenen
88 Zustellserver einen aus. Dabei werden bevorzugt die Zustellserver ausgewählt, bei denen
89 der Empfänger ein X.509 Zertifikat zum Verschlüsseln der Zustellstücke hinterlegt hat. Falls
90 mehrere Zustellserver ein Zertifikat gespeichert haben, wird einer ausgewählt, bei dem der
91 Empfänger keine Einschränkungen hinsichtlich des MIME-Types eingestellt hat. Falls dann
92 immer noch mehrere Zustellserver in der Liste sind, wird, falls eine Liste von bevorzugten
93 Zustellservern in MOA-ZS konfiguriert wurde, einer aus dieser Prioritätsliste ausgewählt.
94 Andernfalls wird einer per Zufall ausgewählt.

95 Sind alle notwendigen Daten vorhanden und kann an den Empfänger elektronisch zugestellt
96 werden, wird der Request in einer Datenbank persistens gespeichert. Ist MOA-ZS als Cluster
97 installiert bzw. konfiguriert worden, wird durch Ermitteln der Anzahl der Request-Objekte mit
98 der jeweiligen `queue_id` die Queue die am wenigsten ausgelastet ist ausgewählt. Wenn das
99 die eigene Queue ist, wird die `ID` des Zustellstücks aus der Datenbank in die Queue
100 eingetragen. Falls die eigene Queue voll ist oder es eine weniger befüllte Queue gibt, wird
101 das Zustellstück mit der `queue_id -1` gespeichert, damit es vom nächsten Watchdog-Thread
102 importiert wird (Siehe auch Kapitel 2.1.4 Watchdog-Thread).

103 War eines der Zustellstücke vom Typ `DocumentReference`, gibt MOA-ZS eine Antwort vom
104 Typ `PartialSuccess` an die Absenderapplikation zurück, ansonsten eine Antwort vom Typ
105 `Success` (Siehe auch Kapitel 2.2 Schnittstellen)

106 1.2 Asynchroner Teil

107 Abbildung 2, „Asynchroner Teil der Requestverarbeitung“ beschreibt den Ablauf im
108 asynchronen Teil von MOA-ZS.



109

110

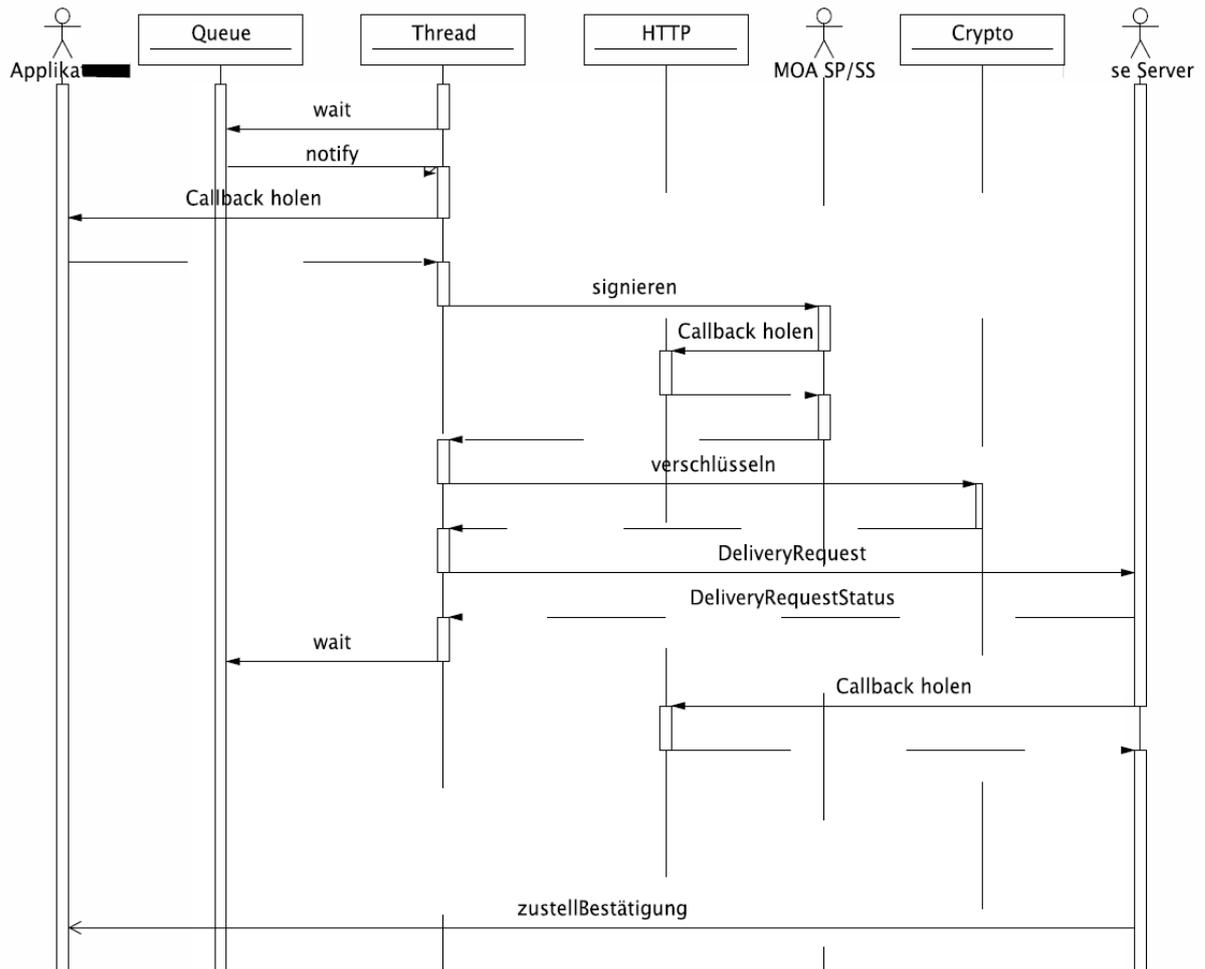
Abbildung 2: Asynchroner Teil der Requestverarbeitung

111

112

113

Abbildung 3, „Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI)“ beschreibt den Ablauf im asynchronen Teil von MOA-ZS wenn eines der mitgeschickten Dokumente durch eine Callback-URI referenziert wird.



114

115

Abbildung 3: Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI)

116

1.2.1 Aufbereiten des Zustellstücks

117

118

119

Im ersten Schritt wird für Zustellstücke ohne XML-Dokument ein XML-Deckblatt erstellt. Dieses enthält die Absender- und Empfängerdaten und eine Liste der angehängten Dokumente.

120

121

122

123

Sind Payload-Elemente vom Typ `DocumentReference` vorhanden, wird versucht, die referenzierten Dokumente via HTTP bzw HTTPs nachzuladen. In der Konfigurationsdatei `moazs_config.xml` kann eingestellt werden, wie oft und mit welchem zeitlichen Abstand das Nachladen versucht werden soll, falls ein Fehler beim Verbindungsaufbau auftritt.

124

125

XML-Dokumente mit einer `XMLProfilID` werden mit den Metadaten, die im Profil eingetragen sind, angereichert.

126

1.2.2 Signieren des Zustellstücks

127

128

129

130

Im nächsten Schritt wird das Zustellstück mit Hilfe von MOA-SS mit einer Amtssignatur versehen. Dieser Schritt wird (wieder konfigurierbar wie oft) wiederholt, falls keine Verbindung zu MOA-SS aufgebaut werden kann. Falls eine Fehlermeldung von MOA-SS retourniert wird, findet keine weitere Wiederholung statt.

131

1.2.3 Erstellung eines S/MIME Containers

132

133

134

Im nächsten Schritt werden die Dokumententeile zu einem MIME-Container oder Zuse-Container zusammengefasst. Welches Format erstellt wird, kann vom Administrator in der Konfiguration eingestellt werden. Falls ein Zertifikat im Zustellserver eingetragen worden ist,

135 wird dieser Container verschlüsselt (RSA, DES_EDE). Das verschlüsselte Zustellstück wird als
136 S/MIME-Container gespeichert.

137 **1.2.4 Übergabe an den Zustellserver**

138 Anschließend wird das Zustellstück an den Zustellserver übergeben. Auch hier können
139 wieder eine maximale Anzahl von Versuchen und eine Zeitspanne zwischen den Versuchen
140 angegeben werden.

141 **1.2.5 Benachrichtigung des Absenders**

142 Im Fall der erfolgreichen Übergabe an einen Zustellserver oder im Fall eines Fehlers, der
143 nicht durch den Ausfall einer Internetverbindung verursacht wurde, wird an den Absender
144 eine Benachrichtigung verschickt (SOAP-Request oder Email). Außerdem wird die Meldung
145 im Logfile verzeichnet und das Requestobjekt inklusive aller Zustellstücke gelöscht.

146 2. Detaillierte Beschreibung von MOA-ZS

147 2.1 Funktionen

148 2.1.1 Vollständigkeitsprüfung

149 Da es sich bei den SOAP-Nachrichten, die mit MOA-ZS verarbeitet werden, nicht um RPC
150 Nachrichten, sondern um Nachrichten vom Typ *document/literal* handelt, kommen keine mit
151 WSDL2JAVA generierten Klassen zum Einsatz, sondern der Body der Nachrichten wird als
152 DOM-Objekt an die Funktion übergeben.

153 Die Prüfung der Pflichtfelder erfolgt mit Hilfe von XPath-Ausdrücken. Dabei werden die
154 statischen Methoden der Klasse `moazs.util.MoaXPath` verwendet. Genauere Details sind
155 in [5] bzw im Quellcode der Klassen enthalten.

156 Mit deren Methode `getFirstString(Element in, String xpath)` wird der erste Knoten
157 der Ergebnisliste des XPath-Ausdrucks in einen String konvertiert und dann zurückgegeben.

```
158 MoaXPath.getFirstString(physicalPerson, "p:Name/p:GivenName/text()")
```

159 Mit diesem Ausdruck wird der Wert des ersten Textknotens unterhalb von `p:GivenName`
160 ermittelt.

161 Mit der Methode `getFirstElement(Element in, String xpath)` wird der erste Knoten
162 der Ergebnisliste des XPath-Ausdrucks als `Element` zurückgegeben. Dieses Element kann
163 dann als Startpunkt für weitere XPath Anfragen verwendet werden.

164 Mit der Methode `getAllElement(Element in, String xpath)` wird die gesamte
165 Ergebnisliste des XPath-Ausdrucks zurückgegeben. Diese Methode kommt vor allem zum
166 Einsatz, um über Blöcke zu iterieren, die in einem Request mehrmals vorkommen können
167 (z.B.: Adressen, Binärdateien, usw.)

168 2.1.2 Lastverteilung

169 Wenn die Anzahl der Zustellstücke in der Queue einen konfigurierbaren Schwellwert
170 übersteigt, startet die Queue einen neuen Workerthread (bis zu einem konfigurierbaren
171 Maximalwert).

172 Die Workerthreads bekommen die Request-ID in der `doSomething`-Methode übergeben
173 und laden das zugehörige Requestobjekt aus der Datenbank. Anhand des Status des
174 Request-Objekts wird eine der Verarbeitungsmethoden ausgewählt.

175 2.1.3 Auswahl des Zustellservers

176 Wenn eine Zusekopf-Anfrage mehrere Zustellserver findet, wird zunächst eine Liste aller
177 Zustellserver erstellt, bei denen der Empfänger ein Zertifikat zur Verschlüsselung der
178 Zustellstücke installiert hat. Falls keiner der Zustellserver ein Zertifikat bereithält oder
179 mehrere Zustellserver ein Zertifikat installiert haben, werden aus den Verbleibenden die
180jenigen gefiltert, die als `MimeType` `"*/"` angegeben haben. Sollten dann immer noch mehr
181als einer überbleiben oder keiner der Zustellserver den `Mimetype` `"*/"` akzeptieren, wird jener
182ausgewählt, welcher in der Konfiguration als bevorzugter eingetragen wurde. Andernfalls
183wird einer per Zufall ausgewählt (siehe Zustellgesetz).

184 Die Auswahl der Zustellserver erfolgt im synchronen Teil eines Request. Dabei verwendet die
185 Methode `getZuseserver` der Klasse `moazs.api.MoaSync` die Methoden der Klasse
186 `moazs.external.ZuseRequest`. Genauere Details sind in [5] bzw im Quellcode der
187 Klassen enthalten.

188 2.1.4 Watchdog-Thread

189 Der Watchdog-Thread schreibt periodisch einen Timestamp in die Tabelle *queue_info* und
190 prüft dann, ob es Einträge anderer Clusterknoten gibt, deren Timestamp eine gewisse Zeit
191 lang nicht aktualisiert worden sind. Falls eine solche Queue gefunden wird, wird versucht,
192 einen HTTP-Request an die in der Tabelle angegeben "Ping"-URL abzusetzen. Wenn dieser
193 Request nicht beantwortet wird, werden alle Einträge in der Request-Tabelle mit der
194 entsprechenden Queue-id auf -1 gesetzt.

195 Im nächsten Durchlauf sammelt der Watchdog-Thread alle Einträge in der Request-Tabelle
196 ein, die die *queue_id* -1 haben und importiert sie in die eigene Queue bis diese voll ist. In
197 einem Cluster versuchen das möglicherweise mehrere Knoten gleichzeitig, der Zugriff ist
198 aber transaktionsgeschützt, sodaß nur der erste Knoten Erfolg hat. Der Watchdog-Thread
199 wird von einem *ServletContextListener* beim Start von MOA-ZS gestartet. Genauere
200 Informationen können in [5] bzw dem Quellcode der Klasse *moazs.watchdog.Watchdog*
201 entnommen werden.

202 2.1.5 Signaturerstellung

203 MOA-ZS signiert optional die Zustellstücke mit Hilfe von MOA-SS. Dabei wird eine XML-
204 Signatur in das XML-Dokument des Zustellstücks bzw in das XML-Deckblatt des
205 Zustellstücks eingebunden. In welchen Knoten die Signatur eingefügt werden soll, kann der
206 Absender mit Hilfe des Signatur-XPath-Ausdrucks im Request festlegen. Die Signatur wird
207 im enveloped-Format erstellt, für alle zusätzlichen Dokumente, die verschickt werden, wird
208 ein *DataObjectInfo* Block angelegt, der das Dokument und eine Referenz auf den
209 lokalen Dateinamen beim Empfänger enthält. Mit diesen Daten erstellt MOA-SS eine
210 Detached-Signatur. Dabei speichert der Signatur-Knoten die HashWerte der Dokumente und
211 das Ergebnis der Signaturberechnung.

212 Clientseitig ist es dadurch möglich, die Signatur des XML-Dokuments unabhängig von der
213 Signatur der mitgeschickten Dokumente zu überprüfen, d.h. es kann die Korrektheit der
214 Signatur auch bei Fehlen eines Dokumentes für den erhaltenen Teil überprüft werden.

215 2.1.6 Verschlüsselung

216 Vor dem Verschlüsseln wird das Dokument in einen MIME-Container oder einen
217 Zusecontainer verpackt. Das Format, das erstellt werden soll, kann mit Hilfe der
218 Konfigurationsdatei *moazs_config.xml* ausgewählt werden. Das resultierende Objekt wird
219 dann mit dem Zertifikat des Users verschlüsselt und in einen SMIME-Container verpackt.

220 Dabei kommt gemäß der Spezifikation des SMIME Standards als symetrischer
221 Verschlüsselungsalgorithmus Triple-DES im CBC-Modus zum Einsatz. Das ist ein Standard,
222 den alle SMIME-Clients unterstützen müssen und der verwendet werden soll, wenn dem
223 Absender nicht bekannt ist, ob der Empfänger eine stärkere Verschlüsselung akzeptiert.

224 2.2 Schnittstellen

225 2.2.1 Applikation an MOA-ZS

226 Die SOAP-Requests, über die die Absenderapplikation mit MOA-ZS kommuniziert,
227 verwenden *DeliveryRequest*, um eine Anfrage zu senden. Die dazugehörige Antwort ist
228 vom Typ *DeliveryResponse*. Der *DeliveryRequest* besteht aus den Elementen *Sender*,
229 *Receiver*, *MetaData*, *XMLDocument* und *Payload*. Dabei ist *XMLDocument* optional und
230 *Payload* kann 0 bis n mal vorkommen.

231 **Sender**

232 Im `Sender`-Element werden die ID des Absenderprofils und der Name der
233 Schlüsselgruppe aus MOA-SS übertragen. Falls das Dokument nicht signiert werden
234 soll, muss der Name der Schlüsselgruppe leer sein.

235

```
236 <Sender>
237   <ProfileID>absender123</ProfileID>
238   <SignatureProfileID>SchluesselgruppenName</SignatureProfileID>
239 </Sender>
```

240

241 **Receiver**

242 Im `Receiver` Element sind die Empfängerinformationen enthalten. Das MOA-ZS
243 Schema verwendet dabei Elemente aus einer simplifizierten Version des `PersonData`
244 2.0 Schemas. Ein Empfänger kann dabei auf mehrere Arten adressiert werden.

- 245 • unverschlüsselte Zustell-bPK des Empfängers
- 246 • verschlüsselte Zustell-bPK des Empfängers
- 247 • bPK des Verfahrensbereichs der Absenderapplikation
- 248 • Name und Geburtsdatum
- 249 • Name und Verständigungsadresse (und Geburtsdatum bei RSa-Zustellung)
- 250 • Name und Postadresse (und Geburtsdatum bei RSa-Zustellung)
- 251 • Stammzahl von nicht natürlichen Personen (z.B. Firmenbuchnummer,
252 Vereinsnummer, usw.)

253 In jedem Fall muß ein `Person`-Element vom Typ `p:PhysicalPerson` oder vom Typ
254 `p:CorporateBody` vorhanden sein. Bei einer natürlichen Person muß ein Feld vom
255 Typ `p:Name` vorhanden sein, bei einer RSa-Zustellung auch ein Feld
256 `p:DateOfBirth`.

257

```
258 <Receiver>
259   <p:PhysicalPerson>
260     <p:Name>
261       <p:GivenName>Max</p:GivenName>
262       <p:FamilyName>Mustermann</p:FamilyName>
263     </p:Name>
264     <p:DateOfBirth>1975-01-15</p:DateOfBirth>
265   </p:PhysicalPerson>
266 </Receiver>
```

267

268 Für eine nichtnatürliche Person werden die Elemente `p:Fullname` und
269 `p:Organisation` (optional) verwendet.

270

```
271 <Receiver>
272   <p:CorporateBody>
273     <p:Fullname>FooBar GmbH<p:Fullname>
274     <p:Organisation>RnD Abteilung</p:Organisation>
275   </p:CorporateBody>
```

276 </Receiver>

277

278 Wenn keine bPK zur Identifikation verwendet wird, muß zusätzlich zu den
279 Personendaten mindestens eine Adresse angegeben werden. Adressen können
280 mehrmals vorkommen und werden in einem Address-Element übergeben, welches
281 vom Typ p:PostalAddress, p:InternetAddress oder p:TelephoneAddress sein
282 kann.

283

284 <Receiver>

285 <p:PhysicalPerson>

286 <p:Identification>

287 <p:Value>A123124123ES...</p:Value>

288 <p:Type>urn:publicid:gv.at:cdid+ZS</p:Type>

289 </p:Identification>

290 </p:PhysicalPerson>

291 <p:InternetAddress>

292 <p:Address>roadrunner@acme.com</p:Address>

293 </p:InternetAddress>

294 </Receiver>

295

296 Eine Adresse vom Typ p:InternetAddress hat genau ein Unterelement vom Typ
297 p:Address, in dem die Emailadresse enthalten ist.

298 Eine Adresse vom Typ p:TelephonAddress enthält ein Unterelement vom Typ
299 p:Number, in dem die Telefonnummer enthalten ist.

300 Eine Adresse vom Typ p:PostalAddress enthält die Elemente p:CountryCode
301 (Ländercode optional), p:PostalCode (PLZ), p:Municipality (Ort) und
302 p:DeliveryAddress. Letzteres enthält dabei die Unterelemente p:StreetName,
303 p:BuildingNumber, p:Unit (Stiege optional) und p:DoorNumber (optional)

304

305 <p:PostalAddress>

306 <p:CountryCode>AT</p:CountryCode>

307 <p:PostalCode>1234</p:PostalCode>

308 <p:Municipality>Sonstwo</p:Municipality>

309 <p:DeliveryAddress>

310 <p:StreetName>Foostraße</p:StreetName>

311 <p:BuildingNumber>42</p:BuildingNumber>

312 <p:Unit>4</p:Unit>

313 <p:DoorNumber>2</p:DoorNumber>

314 </p:DeliveryAddress>

315 </p:PostalAddress>

316

317 **MetaData**

318 Das MetaData-Element enthält die Elemente AppDeliveryID (eine ID, unter der das
319 Zustellstück von der Absenderapplikation gespeichert ist), DeliveryQuality (enthält
320 entweder den String RSa oder nonRSa), das Element RequiresEncryption (vom
321 Typ boolean), DualDelivery (duale Zustellung) und MailBody.

```

322
323     <MetaData>
324         <AppDeliveryID>DOK1234</AppDeliveryID>
325         <DeliveryQuality>non-RSa</DeliveryQuality>
326         <RequiresEncryption>true</RequiresEncryption>
327     </MetaData>

```

328

329 XMLDocument

330 Ein `XMLDocument` Element enthält das Unterelement `XMLContent` vom Typ `xs:any`.
331 In diesem Element kann ein zu verschickendes XML-Dokument eingebettet werden.
332 Weiters wird entweder eine `XMLProfileID` angegeben oder ein `FileName`,
333 `MIMETYPE`, `ResultingMIMETYPE`, `SignatureXPath`, `SignatureStylesheet` und
334 ein `PreviewStylesheet` Element. Dabei ist im `FileName`-Element der Dateiname
335 unter dem das Dokument beim Empfänger gespeichert wird, enthalten. Im `MIMETYPE`
336 Element wird der `Mimetype` des XML-Dokuments gespeichert, im Element
337 `ResultingMIMETYPE` der `Mimety`p nach Anwendung des `PreviewStylesheets` für die
338 Bürgerkartenumgebung des Empfängers. Mit Hilfe des Elements `SignatureXPath`
339 und seinem Attribut `Index` kann spezifiziert werden, unter welchem Knoten die
340 XMLDSIG-Signatur von MOA-SS in das XML-Dokument eingefügt werden soll. Die
341 Elemente `SignatureStylesheet` und `PreviewStylesheet` enthalten jeweils ein
342 Unterelement `XMLContent`, das ebenfalls vom Typ `xs:any` ist und ein `XSLT`-
343 `Stylesheet` enthalten soll. Das `Signatur-Stylesheet` wird vor der Berechnung des
344 Hashwerts für die Signaturerstellung durch MOA-SS angewandt und ist optional. Das
345 `PreviewStylesheet` enthält außerdem ein `FileName` Element, das wie auch beim
346 XML-Dokument den Dateinamen beim Empfänger angibt.

347

```

348 <XMLDocument>
349     <XMLContent>
350         <foo:foodoc xmlns:foo="urn:foons">ich bin ein xmldokument</foo:foodoc>
351     </XMLContent>
352     <FileName>foo.xml</FileName>
353     <MIMETYPE>application/x-foo</MIMETYPE>
354     <ResultingMIMETYPE>text/xhtml</ResultingMIMETYPE>
355     <SignatureXPath Index="0">/foo:foodoc</SignatureXPath>
356     <PreviewStylesheet>
357         <XMLContent>
358             <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSLT"
359             xmlns:foo="urn:foons">
360                 <xsl:template match='foo:foobar'>
361                     <html><body><xsl:value-of select="."/></body></html>
362                 </xsl:tempalte>
363             </xsl:stylesheet>
364         </XMLContent>
365         <FileName>preview.xsl</FileName>
366     </PreviewStylesheet>
367 </XMLDocument>

```

368

369 Payload

370 Ein `Payload` Element enthält entweder ein `BinaryDocument` oder ein `DocumentReference`
371 Element. Im `BinaryDocument` Element werden die Dokumente Base64-codiert übergeben.
372 Dabei kommen die Elemente `Base64Content`, `FileName` und `MIMEType` zum Einsatz.
373 `DocumentReference` enthält die Unterelemente `URL`, `FileName`, `MIMEType` und optional
374 `MD5Checksum`. Das Element `URL` enthält die URL unter der MOA-ZS das Dokument via
375 HTTP oder HTTPS abholen kann.

376

```
377 <Payload>
378   <BinaryDocument>
379     <Base64Content>ASIDOJAISDJ234OIAJD2DDD...<Base64Content>
380     <FileName>bescheid.pdf</FileName>
381     <MIMEType>application/pdf<MIMEType>
382   </BinaryDocument>
383 </Payload>
```

384

```
385 <Payload>
386   <DocumentReference>
387     <URI>http://server/bescheid.pdf<URI>
388     <FileName>bescheid.pdf</FileName>
389     <MIMEType>application/pdf<MIMEType>
390     <MD5Checksum>88ffaa8a88888a8a88a883</MD5Checksum>
391   </DocumentReference>
392 </Payload>
```

393 Dokumente können an MOA-ZS auch mit "Soap with Attachments" (SwA) übergeben
394 werden. In diesem Fall kommt kein `Payload`-Element zum Einsatz, da alle benötigten Daten
395 (Mimetype, Dateiname, ...) als MIME-Header im SwA-Part mitgeschickt werden.

396 2.2.2 MOA-ZS an Applikation

397 Antworten des synchronen Teils an die Applikation erfolgen im `DeliveryResponse`. Dieses
398 Element enthält je nach Antwort ein `Success`, `Error` oder `PartialSuccess` Element.
399 `PartialSuccess` wird verschickt, wenn die Applikation ein Dokument mit `Callback-URI`
400 verschickt hat. Alle drei Elemente haben ein `AppDeliveryID` und `MZSDeliveryID` Element.
401 `MZSDeliveryID` ist die ID des Zustellstücks in MOA-ZS, `AppDeliveryID` die ID des
402 Zustellstücks der Absenderapplikation. Im Fehlerfall kommen auch noch das Element `Code`
403 und `Text`. Dort werden eine Fehlernummer und eine textuelle Beschreibung eingetragen.

404 Wenn im asynchronen Teil eine Meldung an die Applikation verschickt werden muß, kommt
405 eine Nachricht vom Typ `DeliveryNotification` zum Einsatz, deren Aufbau ähnlich ist wie
406 der von `DeliveryResponse`. Im Fall einer Meldung über die Übergabe des Zustellstücks an
407 einen Zustellserver enthält es zusätzlich das Element `DeliveryStatement` mit den
408 Unterlementen `DeliveryServer`, `Timestamp` und `ZSDeliveryID`.

409 2.2.3 Zusekopf

410 Die Schnittstelle des *Zusekopf* ist im Detail in [3] beschrieben.

411 2.2.4 Stammzahlenregister

412 Die Schnittstelle zum SZR ist im Detail in [2] im Kapitel *Prüfung der Adressierbarkeit*
413 beschrieben.

414 **2.2.5 MOA-SS**

415 Die Schnittstelle zu MOA-SS ist im Detail in [2] im Kapitel „*Anbringen der Absendersignatur*“
416 beschrieben.

417 **2.2.6 Zustellserver**

418 Die Schnittstelle des *Zustellserver* ist im detail in [4] beschrieben.

419 **2.3 Datenorganisation**

420 MOA-ZS speichert alle Requestdaten in einer relationalen Datenbank. Die Daten werden
421 allerdings nur solange in der Datenbank gespeichert, bis der Request abgearbeitet ist, dann
422 werden sie gelöscht. Die Datenbank wird also verwendet, um Transaktionsicherheit und
423 einen geordneten Wiederanlauf nach einem Systemausfall zu gewährleisten.

[1]	MOA-ZS Administrationshandbuch
[2]	MOA-ZS Technische Dokumentation
[3]	Elektronische Zustellung – Zustellkopf – Schnittstellenspezifikation 1.3.0
[4]	Elektronische Zustellung – Message Spezifikation 1.3.0
[5]	MOA-ZS Javadoc – docs/apidocs/index.html
[6]	Bouncycastle JCE Provider – http://www.bouncycastle.org
[7]	Apache AXIS - http://ws.apache.org/axis/
[8]	Jakarta HTTP Components - http://hc.apache.org
[9]	JConfig – http://www.jconfig.org

3. Historie

Version 0.1	Datum 01.11.2004	Kommentar Erstellung
Ersteller Nikolaus Gradwohl		
Version 0.11	Datum 04.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.2	Datum 05.11.2004	Kommentar Einarbeitung Revisionssergebnisse
Ersteller Nikolaus Gradwohl		
Version 0.21	Datum 07.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.3	Datum 10.11.2004	Kommentar Einarbeitung Revisionsergebnisse
Ersteller Nikolaus Gradwohl		
Version 0.31	Datum 10.11.2004	Kommentar Revision
Ersteller Andreas Erlacher		
Version 0.4	Datum 10.11.2004	Kommentar Einarbeitung Revisionsergebnisse
Ersteller Nikolaus Gradwohl		
Version 1.1.0	Datum 07.11.2008	Kommentar Anpassungen an die MOA-ZS Version 1.1.0
Ersteller Arne Tauber		

428 **4. Anhang**

429 **4.1 Begriffe und Abkürzungen**

Begriff, Abkürzung	Beschreibung
MOA-ZS	MOA Zustellservice
MOA-SS/SP	MOA Signaturservice/Signaturprüfung
DOM	Document Object Model
bPK	Bereichsspezifisches Personenkennzeichen
zbPK	Bereichsspezifisches Personenkennzeichen für den Bereich Zustellung

430 **Tabelle 4-1 Begriffe und Abürzungen**

431 **4.2 Tabellenverzeichnis**

432 Tabelle 9-1 Begriffe und Abürzungen..... 18
433

434 **4.3 Abbildungsverzeichnis**

435 Abbildung 1: Synchroner Teil der Requestverarbeitung 4
436 Abbildung 2: Asynchroner Teil der Requestverarbeitung 6
437 Abbildung 3: Asynchroner Teil der Requestverarbeitung (Zustellstück mit Callback-URI) 7
438 Abbildung 4: Datenmodell MOA-ZS **Error! Bookmark not defined.**
439
440

441 **4.4 Konfigurationsdatei moazs_config.xml**

442 TBD - Hier beispielhafte Konfigurationsdatei eintragen

443