

Dokumentation Zustellproxy

Allgemeine E-Government Infrastruktur

Version 1.0, 18.07.2006

DI Arne Tauber – arne.tauber@egiz.gv.at

Zusammenfassung:

MOA-ZS bietet zur Verständigung über den Zustellerfolg per se keine direkte synchrone Auskunft an die zustellende Applikation. Grundsätzlich wird die Applikation nur über die erfolgreiche bzw. erfolglose Annahme des Zustellstücks durch MOA-ZS selbst informiert. Das hier dokumentierte MOA-ZS Proxy Modul hat die Aufgabe, die asynchronen Zustellbestätigungen über ein Webservice anzunehmen und über das MOA-ZS Protokoll der Applikation weiterzureichen. Da das MOAZS-Proxy Modul das bestehende MOAZS SOAP Interface implementiert, kann dieses Modul wie jeder MOAZS Server angesprochen werden, mit dem Unterschied, dass an die Stelle einer asynchronen Zustellbenachrichtigung eine (semi-) Synchrone tritt

Inhaltsverzeichnis:

1.1	Grundlagen	3
1.2	Funktionsbeschreibung	4
1.3	Voraussetzungen zur Nutzung der Anwendung	5
2.1	Zustellstück übergeben	6
2.2	DeliveryNotification	7
2.3	Abfrage von DeliveryNotification Nachrichten	7
3.1	Systemanforderungen	10
3.2	Installation	10
3.3	Konfiguration	10

Anmerkung: Zur besseren Lesbarkeit wurde in diesem Dokument teilweise auf geschlechtsspezifische Formulierungen verzichtet. Die verwendeten Formulierungen richten sich jedoch ausdrücklich an beide Geschlechter.

Revision History

Version	Datum	Autor(en)	
1.0	18.07.2006	Arne Tauber	Initialversion

1 Kurzbeschreibung

1.1 Grundlagen

MOA-ZS bietet zur Verständigung über den Zustellerfolg per se keine direkte synchrone Auskunft an die zustellende Applikation. Grundsätzlich wird die Applikation nur über die erfolgreiche bzw. erfolglose Annahme des Zustellstücks durch MOA-ZS selbst informiert. Obschon diese Information auf Basis der aus dem Zustellkopf erfragten Informationen basiert, ist gerade die direkte Erfolgsmeldung über den tatsächlichen Zustellerfolg von besonderem Interesse. MOA-ZS bietet zwei asynchrone Mechanismen für die Übermittlung des Zustellstatus an die Applikation. Dies kann entweder per E-mail oder über eine SOAP Schnittstelle (WebService) erfolgen. Ohne weitere Mechanismen sind diese asynchronen Statusübermittlungen jedoch nicht von direktem Nutzen, da entweder die Applikation selbst das Webinterface implementieren muss (über SOAP), bzw. es leicht Probleme geben kann bei vielen Zustellungen und einer asynchronen Benachrichtigung über E-mail.

Der Mechanismus des MOAZS-Proxy Moduls besteht darin, die asynchronen Zustellbestätigungen über ein Webservice anzunehmen und über das MOA-ZS Protokoll der Applikation weiterzureichen. Da das MOAZS-Proxy Modul das bestehende MOAZS SOAP Interface implementiert, kann dieses Modul wie jeder MOAZS Server angesprochen werden, mit dem Unterschied, dass an die Stelle einer asynchronen Zustellbenachrichtigung eine (semi-) Synchrone tritt. Der Begriff „semi“ wird deshalb verwendet, weil eine vollkommen synchrone Bestätigung in dem Sinne nicht existiert.

Sofern keine Massenzustellung stattfindet, sodass in der Folge alle Zustellstücke am MOAZS Server „gespoolt“ werden, ist die Wahrscheinlichkeit relativ hoch, dass das Zustellstück rasch bzw. direkt am Zustellserver angenommen wird.

Bei Anomalien die bis zur Kontaktaufnahme des Zustellkopfs auftreten, wird der Client ohnehin mit einer Fehlermeldung benachrichtigt. Der MOAZS-Proxy kann daher mit einem „Timeout“ konfiguriert werden, innerhalb dessen er versucht den Zustellstatus zu ermitteln. Ist dieser Timeout vernünftig konfiguriert, kann der Status für die meisten Zustellstücke bereits der Applikation mitgeteilt werden. Dies entspricht einer synchronen Benachrichtigung. Kann das Zustellstück jedoch z.b. temporär nicht zugestellt werden und läuft der Timeout des MOAZS-Proxy Moduls ab, ist der Zustellstatus zu dem Zeitpunkt noch unbekannt. In diesem Fall muss die Clientapplikation jedoch wieder über einen asynchronen Mechanismus den Zustellstatus ermitteln. Da jedoch der Zustellstatus in jedem Fall vom vom Webservice entgegengenommen und archiviert wurde, kann die Applikation jederzeit versuchen, den Status über ein Webinterface oder über eine Java API abzufragen. Für diesen Zweck wurde ein eigenes SOAP Interface entwickelt.

1.2 Funktionsbeschreibung

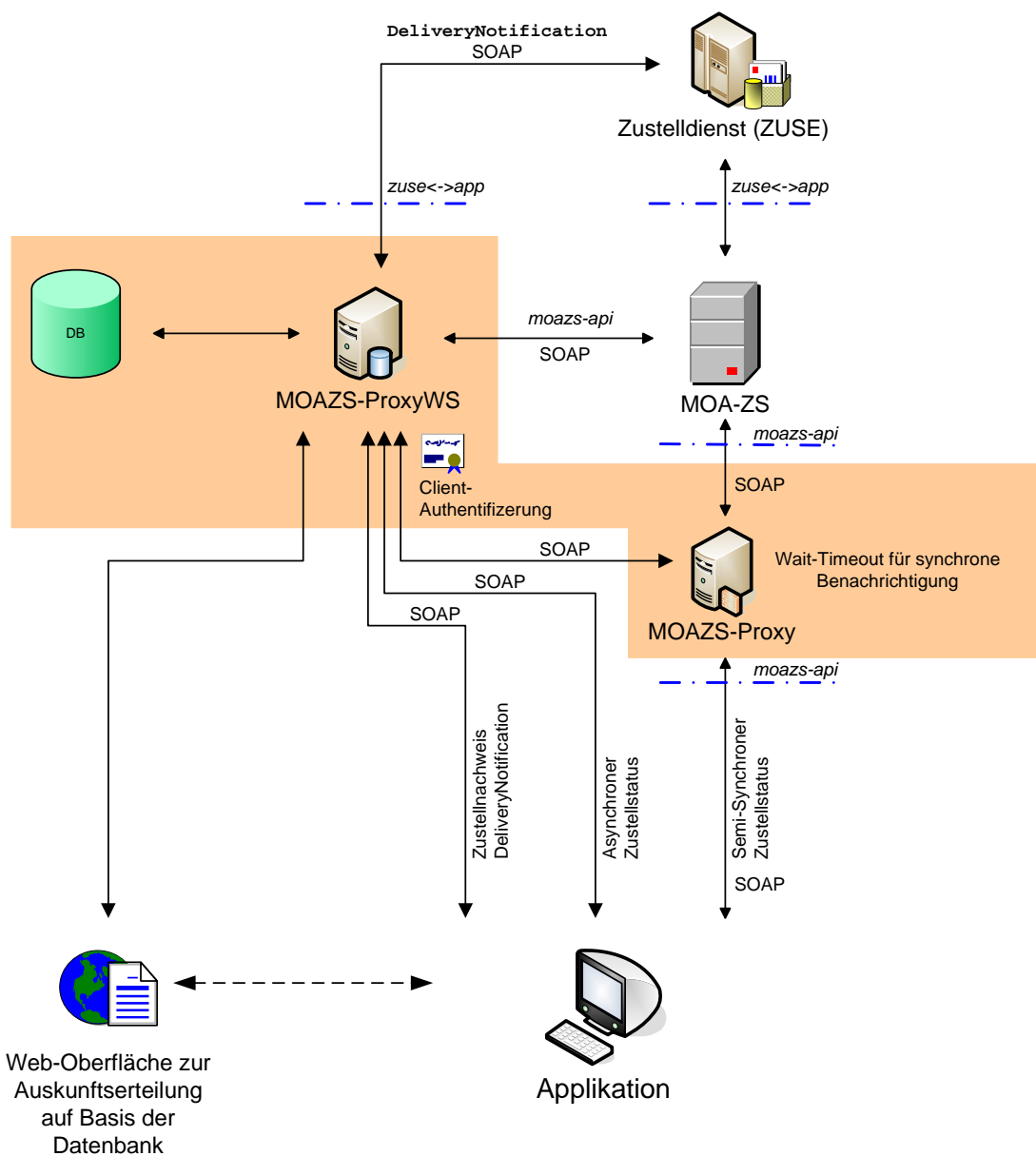


Abbildung 1: Prozessmodell des MOA-ZS Proxy Moduls

Das Modul besteht aus 2 Komponenten. Dem MOA-ZS Proxy selbst, der das MOA-ZS Protokoll implementiert und Zustellungen an MOA-ZS weiterleitet und einem Webservice, das die DeliveryNotification Nachrichten von MOA-ZS bzw. von Zustellservern entgegennimmt.

Um die semi-synchrone Zustellung nutzen zu können, muss eine Applikation nicht mehr direkt über MOA-ZS kommunizieren, sondern kann dafür den MOA-ZS Proxy verwenden. Es sind diesbezüglich keine Änderungen notwendig, da der Proxy dasselbe Protokoll verwendet und implementiert wie MOA-ZS. Daher erfolgt die Zwischenschaltung des Proxys vollständig transparent.

Das Proxy Modul implementiert zusätzlich ein Webservice (MOA-ZS-ProxyWS) und eine Datenbank, um folgende Statusnachrichten empfangen und abspeichern zu können:

1. mzs:DeliveryNotification: diese Nachricht beinhaltet den Zustellstatus seitens MOA-ZS. Dabei wird unterschieden zwischen
 - a.Success: das Zustellstück konnte erfolgreich dem Zustellserver übergeben werden
 - b.Error: es ist ein Fehler bei der Zustellstückübergabe an den Zustellserver aufgetreten.
2. zs:DeliveryNotification: im Falle von RSa Sendungen, werden Zustellbestätigungen in Form dieser Nachricht an das Webservice geschickt.

Der Zustellprozess über den MOA-ZS Proxy gestaltet sich wie folgt:

1. Die Applikation übergibt dem MOA-ZS Proxy über das MOA-ZS Protokoll ein Zustellstück.
2. Der MOA-ZS Proxy reicht diesen Zustellstückrequest an MOA-ZS 1:1 weiter.
3. Antwortet MOA-ZS mit einem Fehler (z.B. Empfänger nicht gefunden oä.), wird dieser Fehler direkt an die Applikation weitergereicht. Die Zustellung bricht hiermit ab.
4. Antwortet MOA-ZS mit Success oder PartialSuccess, d.h. das Zustellstück wurde von MOA-ZS übernommen, wartet der MOA-ZS Proxy über sein Webservice eine bestimmte Zeit (konfigurierbar) auf eine Zustellbestätigung seitens MOA-ZS. Dabei können 3 Fälle unterschieden werden:
 - a.Innerhalb der Zeitspanne wird eine Success Meldung empfangen: es wird an die Applikation eine Success Meldung zurückgeliefert. Die Zustellung erfolgt somit synchron.
 - b.Innerhalb der Zeitspanne wird eine Error Meldung empfangen: es wird an die Applikation eine Error Meldung zurückgeliefert. Die Zustellung erfolgt somit synchron.
 - c.Innerhalb der Zeitspanne wird keine entsprechende Meldung empfangen: an die Applikation wird eine PartialSuccess Meldung zurückgeliefert. Dies signalisiert eine semisynchrone Zustellung.

Alle Typen von DeliveryNotification (Zustellstati, Zustellnachweise) werden für eine bestimmte konfigurierbare Zeit in der Datenbank abgelegt und können somit jederzeit über eine Java API oder über ein SOAP Interface abgefragt werden.

1.3 Voraussetzungen zur Nutzung der Anwendung

- JDK 1.4
- Application Server (z.B. Tomcat)
- Datenbank (z.B. MySQL)

2 Anwendungsbeschreibung

Der nachfolgenden Anwendungsbeschreibung wird folgende Terminologie bzw. werden folgende Parameter zu Grunde gelegt:

Parameter

Name	Wertebereich	Beschreibung
%CONTEXT%		Der Application-Kontext und dem die Applikation betrieben wird. Z.B. http://mein-domain.com/zsproxy/

Der MOA-ZS Proxy implementiert folgende 3 Webservice Interfaces:

Name	Beschreibung
DeliveryRequest	Dieses Webservice muss für die Zustellstückübernahme verwendet werden. Es implementiert das MOA-ZS Protokoll und kann somit transparent an Stelle dessen verwendet werden. URL: %CONTEXT%/services/DeliveryRequest
DeliveryNotification	Dieses Webservice nimmt Zustellstati (mzs:DeliveryNotification) von MOA-ZS und Zustellnachweise (zs:DeliveryNotification) von Zustellservern an und speichert diese in der Datenbank. URL: %CONTEXT%/services/DeliveryNotification
DeliveryNotificationRequest	Über dieses Webservice können in der Datenbank abgelegte DeliveryNotification Nachrichten abgefragt werden. URL: %CONTEXT%/services/DeliveryNotificationRequest

2.1 Zustellstück übergeben

Anstelle des MOA-ZS Webservices kann das MOA-ZS Proxy Webservice verwendet werden. Da beide Webservices dasselbe Protokoll verwenden, ist die Zwischenschaltung des Proxys vollkommen transparent.

Beispiel:

MOA-ZS URL: <http://meinebehoerde.gv.at/moazs/services/DeliveryRequest>

Ist der Proxy im selben Applicationcontainer installiert können Sie z.B. die obige URL durch die folgende ersetzen:

Proxy-URL: <http://meinebehoerde.gv.at/zsproxy/services/DeliveryRequest>

2.2 DeliveryNotification

Die Adresse, an welche Zustellstati und Zustellnachweise versendet werden, wird in den MOA-ZS Senderprofilen eingestellt. Tragen Sie dort die Adresse des Proxy DeliveryNotification Webservices ein:

Beispiel eines MOA-ZS Senderprofils (Ausschnitt aus der MOA-ZS Konfigurationsdatei):

```
<category name="zsproxy">
  <fullName>Meine Testbehoerde</fullName>
  <oid>1</oid>
  <organization>keine</organization>
  <postalCode>1234</postalCode>
  <countryCode>AT</countryCode>
  <municipality>Wien</municipality>
  <streetName>Mustergasse</streetName>
  <buildingNumber>1</buildingNumber>
  <unit>a</unit>
  <webserviceUrl>
    http://meinebehoerde.gv.at/zsproxy/services/DeliveryNotification
  </webserviceUrl>
  <SkipDeckblatt>yes</SkipDeckblatt>
</category>
```

Anmerkung: wenn Sie DeliveryNotification Nachrichten von Zustellservern empfangen möchten, müssen Sie darauf achten, dass das Webservice vom Internet aus zugänglich ist.

2.3 Abfrage von DeliveryNotification Nachrichten

Die Abfrage von Zustellstati bzw. Zustellnachweisen erfolgt über das DeliveryNotificationRequest Webservice oder über die Java API.

Ein Zustellstück sollte beim Absenden von der Applikation eine eindeutige ID zugewiesen bekommen. Diese ID wird auch als *AppDeliveryID* bezeichnet. Jede DeliveryNotification ist daher mit einer eindeutigen AppDeliveryID verknüpft. Sowohl bei Abfragen über das Webservice, als auch bei Abfragen über die Java API muss daher die AppDeliveryID als Parameter mitübergeben werden.

2.3.1 Abfrage über die Java API

Verwenden Sie diesbezüglich die Klasse `at.gv.egiz.zsproxy.api.DBAPI`. Diese Klasse enthält statische Methoden, mit der direkt auf die Datenbank zugegriffen werden kann.

Achtung: zuvor muss jedoch der Zugriff auf die Datenbank konfiguriert werden (Host, Username, Passwort, etc.). Wie Hibernate diesbezüglich konfiguriert wird, kann im Abschnitt 3.3 nachgelesen werden.

Folgende statische Methoden stehen zur Verfügung:

```
public static DeliveryNotificationData getMOAZSDeliveryNotification(String appDeliveryId);
public static ZSDeliveryNotification getZSDeliveryNotification(String appDeliveryId);
```

2.3.2 Abfrage über das Webservice

2.3.2.1 Interface

Beispielrequest:

```
<zsp:DeliveryNotificationRequest
xmlns:zsp="http://www.egiz.gv.at/namespaces/zsproxy/20060714#">

  <zsp:AppDeliveryID type="moazs">123456</zsp:AppDeliveryID>
</zsp:DeliveryNotificationRequest>
```

Obiger Request zeigt die XML Struktur, die an das Webservice gesendet werden muss, um einen Status bzw. einen Zustellnachweis abzufragen. Einziger Unterschied zwischen einer DeliveryNotification von MOA-ZS und einem Zustellserver ist das „type“ Attribut des AppDeliveryID Elements. Verwenden Sie folgende Typen:

1. „moazs“ für eine Abfrage eines Zustellstatus von MOA-ZS
2. „zuse“ für eine Abfrage eines Zustellnachweises von einem Zustellserver

Falls ein Eintrag für die angegebene AppDeliveryID gefunden wurde, so wird das DeliveryNotification Element eingebettet zurückgegeben:

Beispiel einer MOA-ZS DeliveryNotification:

```
<zsp:DeliveryNotificationResponse
xmlns:zsp="http://www.egiz.gv.at/namespaces/zsproxy/20060714#">

  <mzs:DeliveryNotification>...</mzs:DeliveryNotification>
</zsp:DeliveryNotificationResponse>
```

Falls kein Eintrag für die angegebene AppDeliveryID gefunden wurde, wird ein NotFound Element zurückgeliefert:

```
<zsp:DeliveryNotificationResponse
xmlns:zsp="http://www.egiz.gv.at/namespaces/zsproxy/20060714#">

  <zsp:NotFound/>
</zsp:DeliveryNotificationResponse>
```

2.3.2.2 Abfrage über Java SOAP Client

Im Package wird auch ein SOAP Client für die Abfrage von DeliveryNotification Nachrichten mitgeliefert.

Der Client ist in der Java Klasse `at.gv.egiz.zsproxy.api.ZSProxyClient` implementiert.

Die Abfrage kann über folgende Methode gestartet werden:

```
public Element getDeliveryNotification(String address, String type, String
appDeliveryId) throws ClientException;
```

Folgende Parameter sind dabei zu verwenden:

1. „address“: bezeichnet die Webservice Adresse des DeliveryNotificationRequest Services des MOA-ZS Proxys.

2. „type“ bezeichnet den Typ der abzufragenden DeliveryNotification. Muss entweder „moazs“ oder „zuse“ sein.
3. „appDeliveryId“ bezeichnet die AppDeliveryID, für welche die DeliveryNotification abgefragt wird.

Beispielcode zur Verwendung:

```
// BEGIN CODE

import org.w3c.dom.Element;
import asit.common.Utills;
import at.gv.egiz.zsproxy.api.ZSProxyClient;

public class TestProxyClient {

    public static void main(String[] args) throws Exception {
        ZSProxyClient client = new ZSProxyClient();
        Element ret =
client.getDeliveryNotification("http://localhost:8080/zsproxy/services/DeliveryNoti
ficationRequest", ZSProxyClient.ZUSE, "app-id-123");
        Utills.serializeElement(ret, System.out);
    }
}
// END CODE
```

Der obige Beispielcode zeigt ein Beispiel eines Request an das DeliveryNotificationRequest Webservice, wobei das Result Element (Entweder die DeliveryNotification oder zsp:NotFound) auf den Standardoutput ausgegeben wird.

3 Deployment

3.1 Systemanforderungen

Es gelten folgende Anforderungen an die Installationsplattform bzw. an die Client-komponenten (die angegebenen Versionen entsprechen der getesteten Umgebung):

3.1.1 Server-Komponenten

- JDK 1.4
- Application Server (getestet mit Tomcat 5.0.28)
- Datenbank mit Java Connector (getestet mit MySQL 4.1)

3.1.2 Client-Komponenten

- SOAP Client
- Allenfalls JDK 1.4, falls API verwendet wird

3.2 Installation

Im Package ist die Datei zsproxy.war mitgeliefert, die üblicherweise in einem Application Server automatisch installiert werden kann.

3.3 Konfiguration

Für die Inbetriebnahme muss eine Konfigurationsdatei angegeben werden. Es gibt 2 Möglichkeiten diese einzubinden:

1. Die Datei muss „zsproxy_config.xml“ benannt werden und den Classpath gegeben werden (z.B. WEB-INF/classes Verzeichnis).
2. Es wird die Systemvariable „zsproxy.configuration“ gesetzt. Der Inhalt dieser Variable muss eine Pfadname sein, der auf die Konfigurationsdatei zeigt.

Anmerkung: Es wird Methode 2 empfohlen, da bei einem Update z.B. der gesamte Webcontainer gelöscht und durch das neue war-File ersetzt werden kann, ohne dabei die Konfigurationsdatei zu löschen.

3.3.1 Konfigurationsdatei

Die hier angeführten Parameter sind unbedingt vor Inbetriebnahme der Anwendung anzupassen und stellen somit eine Minimalkonfiguration dar.

general

Property	Default-Wert	Beschreibung
delete.fetch	false	Gibt an, ob der Eintrag nach eine Abfrage von der Datenbank gelöscht werden soll.
max.backup	0	Gibt die Zeit in Minuten an, wie lange eine DeliveryNotification in der Datenbank gehalten werden soll. Eine Zeit von 0 heisst „für immer“.

moazs

Property	Default-Wert	Beschreibung
connection.url		MOA-ZS Webservice URL, an welche die Requests weitergeleitet werden.
max.retry	10000	Gibt die Zeit in Millisekunden an, wie lange versucht werden soll ein Zustellstatus aus der Datenbank zu ermitteln, bevor die Response an die Applikation zurückgesendet wird.
retry.timeout	1000	Gibt das Intervall an, mit der die Abfragen in der Datenbank stattfinden sollen.

Hibernate

Property	Bedeutung	Beschreibung
hibernate.dialect	Einstellung der verwendeten Datenbank	Siehe hibernate.org.
hibernate.connection.url	Angabe der Verbindung zum Datenbanksystem, incl. des Namens der zu verbindenden Datenbank	JDBC Connection URL
hibernate.connection.charSet	Verwendeter Zeichensatz	Gibt das Intervall an, mit der die Abfragen in der Datenbank stattfinden sollen.
hibernate.connection.driver	JDBC Treiber Klasse	
hibernate.connection.username	Benutzername	
hibernate.connection.password	Passwort	

Für eine genauere Erklärung der Parameter bzw. weitere Parameter siehe <http://www.hibernate.org>.

3.3.2 Log4j Konfiguration

Die MOA-ZS Proxy Applikation verwendet Log4J, wobei der Logger jeder Klasse mit dem Präfix „at.gv.egiz.zsproxy“ versehen ist. Ein Beispielkonfigurationsdatei für Log4J ist im Package enthalten.

3.3.3 Initialisieren der Datenbank

Um die Tabellen für die Datenbank initialisieren zu können, muss zuerst eine Datenbank mit Benutzername und Passwort (wie in obiger Hibernate Konfiguration angegeben) angelegt werden.

Die Datenbank kann anschliessend über eine Weboberfläche initialisiert werden:

Die URL dafür lautet: %CONTEXT%/DatabaseManager

z.B. <http://localhost:8080/zsproxy/DatabaseManager>

Diese Resource ist über den Realm „moa-admin“ abgesichert. Daher muss unter Umständen zuerst ein gültiger Benutzer für diesen Realm eingerichtet werden.

MOA-ZS Proxy Datenbankverwaltung

Tabellen löschen

Tabellen erzeugen

hibernate.c3p0.timeout	100
hibernate.connection.driver_class	com.mysql.jdbc.Driver
hibernate.c3p0.max_statements	0
hibernate.c3p0.max_size	100
hibernate.dialect	org.hibernate.dialect.MySQLDialect
hibernate.c3p0.idle_test_period	100
hibernate.c3p0.min_size	10
hibernate.connection.username	zsproxy
hibernate.c3p0.acquire_increment	1
hibernate.connection.url	jdbc:mysql://localhost/zsproxy
hibernate.connection.password	zsproxy
hibernate.connection.charSet	utf-8