Technische Schnittstellen-beschreibung

Version 2.8.0 – 15.09.2017

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Einleitung

Der Mobilitäts Daten Marktplatz (MDM) hat zum Ziel, den Datenaustausch zwischen Datengebern und Datennehmern mit Hilfe von Schnittstellen zu unterstützen und stellt gleichzeitig ein zentrales Portal mit den gesammelten Informationen über verfügbare Online-Verkehrsdaten einzelner Datengeber dar. Auf diese Weise ermöglicht der MDM seinen Nutzern das Anbieten, Finden und Abonnieren verkehrsrelevanter Online-Daten, ohne dass eine langwierige Suche nach den relevanten Daten und eine aufwendige technische und organisatorische bilaterale Abstimmung zwischen Datennehmern und Datengebern notwendig werden. Der Datenaustausch wird über standardisierte Schnittstellen abgewickelt. Im Ergebnis sollen so die Geschäftsprozesse für alle Beteiligten vereinfacht und die Potentiale vorhandener Datenquellen erschlossen werden.

Diese Schnittstellenbeschreibung wendet sich an potentielle Datengeber und Datennehmer. Kenntnisse in der Implementierung und im Betrieb von SOAP–Webservices [SOAP] bzw. HTTPS-Client/Server-Architekturen werden zur Nutzung der Schnittstellen des MDM-Systems vorausgesetzt.

Über die von der MDM-Plattform angebotenen Schnittstellen können Datengebersysteme und Datennehmersysteme auf die Dienste der Plattform zugreifen. Diese Dienste zur Datenabholung bzw. -anlieferung werden unter definierten, vereinheitlichten URLs angeboten [URL] und erfordern eine zertifikatsbasierte Clientauthentisierung via HTTPS [HTTPS]. Für diese Clientauthentisierung kommen X.509-konforme Zertifikate zum Einsatz [PKI], die vom Betreiber der MDM-Plattform ausgestellt werden. Die Datenübertragung zwischen der MDM-Plattform und den Datengeber- bzw. Datennehmersystemen kann wahlweise über SOAP-basierte Webservices oder einfache HTTPS-GET/POST-Requests erfolgen. Zusätzlich wird die Übertragung per OTS 2- und OCIT-C-Protokoll angeboten.

Bei der Datenübermittlung zwischen MDM-Plattform und Datengebersystemen werden sowohl GZIP-encodierte (d.h. komprimierte) HTTPS-Requests und –Responses als auch unkomprimierte unterstützt. Die Datenübermittlung zwischen MDM-Plattform und Datennehmersystemen findet immer mittels GZIP-encodierter HTTPS-Requests und –Responses statt. Wird SOAP zur Übermittlung verwendet, müssen außerdem die WS-Security Standards eingehalten werden. Dies beinhaltet eine Übertragung des Security-Tokens und ggf. die Signierung der Nachricht.

1.2 Gliederung des Dokuments

Das Dokument ist in die folgenden Abschnitte gegliedert:

- Abschnitt 1 enthält eine kurze Übersicht sowie die referenzierten Dokumente sowie das Abkürzungsverzeichnis.
- In Abschnitt 2 werden die Komponenten des MDM-Systems vorgestellt.
- Abschnitt 3 behandelt die verfügbaren Datenformate.
- Die Schnittstellen der MDM-Plattform für die M2M-Kommunikation werden in Abschnitt 4 beschrieben.
- Abschnitt 4.6 beschreibt die Maßnahmen, mit der die M2M-Kommunikation abgesichert wird.
- Abschnitt 6 zeigt mögliche Meldungen auf, die bei fehlerhaften Requests an die Schnittstellen auftreten können.
- Abschnitt 7 enthält XML-Beispiele für SOAP- und HTTPS-Requests im DATEX II- und im Containerformat sowie ein Beispiel für die Nutzung von OTS 2.

1.3 Referenzierte Dokumente

[Quelle]	Herausgeber
[BHB]	MDM-Benutzerhandbuch, V1.2 http://service.mdm-portal.de/doc/MDM-Benutzerhandbuch.pdf
[DatexIIPSM]	DATEX II V2.0 Exchange Platform Specific Model
[DatexIIPull]	DATEX II V2.0 Pull wsdl
[DatexIIPush]	DATEX II V2.0 Push wsdl
[DatexIISchema]	DATEX II XML Schema 2.0
[DatexIISDG]	DATEX II v2.0 Software Developers Guide, Version v.1.2
[DatexIISpec]	Umfasst die folgenden Dokumente, die auf http://www.datex2.eu allen registrierten Benutzern zum Download bereitstehen: [DatexIIPSM], [DatexIISDG], [DatexIIUserGuide]
[DatexIIUserGuide]	DATEX II v2.0 User Guide v.1.2
[GZIP]	RFC 1952 (Mai 1996) GZIP File Format Specification Version 4.3, http://tools.ietf.org/rfc/rfc1952.txt

[Quelle]	Herausgeber
[HTTP/1.1]	RFC 2616 (Juni 1999) Hypertext Transfer Protocol -- HTTP/1.1 http://www.ietf.org/rfc/rfc2616.txt
[HTTPS]	RFC 2818 (Mai 2000) HTTP over TLS http://www.ietf.org/rfc/rfc2818.txt
[MCS]	MDM Containerformat Spezifikation http://www.mdm-portal.de
[OCIT-C]	OCIT-C Spezifikation Version 1.1_R1 vom 30.10.2014 http://www.ocit.org/OCIT-C.htm
[OTS2]	OTS 2 Spezifikation OTS-Kommunikation Version 02-02-09 http://www.opentrafficsystems.org
[OTS2DIN]	DIN SPEC 91213-1 Open Traffic Systems – OTS 2-Schnittstellenspezifikation – Teil 1: Einführende Erläuterungen für Entscheidungsträger; Januar 2011 DIN SPEC 91213-2 Open Traffic Systems – OTS 2-Schnittstellenspezifikation – Teil 2: Technische Spezifikation für Implementierer; Februar 2011
[PKI]	RFC 2459 (Januar 1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile http://www.ietf.org/rfc/rfc2459.txt
[SOAP]	SOAP Version 1.2 http://www.w3.org/TR/soap12-part1/
[URL]	RFC 1738 (Dezember 1994) Uniform Resource Locators (URL) http://www.ietf.org/rfc/rfc1738.txt
[X.509v3]	ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection – The Directory: Authentication Framework, June 1997. http://www.itu.int/rec/T-REC-X.509-199708-S/en

Tabelle 1: Referenzierte Dokumente

1.4 Abkürzungsverzeichnis

Abkürzung	Auflösung
BASE64	BASE64 beschreibt ein Verfahren zur Kodierung von 8-Bit-Binärdaten in eine Zeichenfolge, die nur aus lesbaren Codepage-unabhängigen ASCII-Zeichen besteht.

Abkürzung	Auflösung
BASt	Bundesanstalt für Straßenwesen
DE	Deutsch
GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IIS	Microsoft Internet Information Services
IT	Informationstechnik/Informationstechnologie
JSSE	Java Secure Socket Extension
M2M	Machine-to-Machine
MDM	Mobilitätsdatenmarktplatz
MDV	Metadatenverzeichnis
OCIT	Open Communication Interface for Road Traffic Control Systems
OTS	Open Traffic Systems
PAS	Publicly Available Specification
PKI	Public Key Infrastructure
PSM	Platform Specific Model
RC	Release Candidate
RFC	Request for Comments
SDG	Software Developers Guide
SNI	Server Name Indication
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator
UTF	UCS Transformation Format
WS	Webserver
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

Tabelle 2: Abkürzungsverzeichnis

2 Komponenten der MDM-Plattform im Überblick

Die MDM-Plattform setzt sich aus vier Komponenten zusammen, die jeweils unterschiedliche Aufgaben übernehmen.

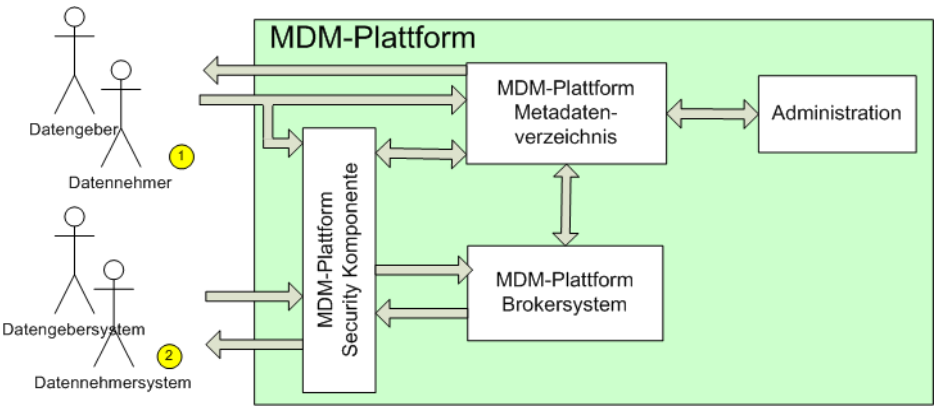

Abbildung 1: Komponenten der MDM-Plattform

Komponente	Beschreibung
Securitykomponente	Über die Securitykomponente können Datennehmersystem/Datengebersystem authentisiert werden, um Dienste nutzen zu können.
Metadatenverzeichnis	Das Metadatenverzeichnis (MDV) dient zur Verwaltung aller für die MDM-Plattform relevanten Informationen und stellt eine Reihe organisatorischer Dienste zur Verfügung.
Brokersystem	Das Brokersystem übernimmt die eigentliche Verarbeitung der Datenpakete und steht daher im Mittelpunkt dieser Schnittstellenbeschreibung.
Administration	Die Administration wird mittels einer webgestützten Benutzeroberfläche (GUI) realisiert, siehe [BHB]

Tabelle 3: Übersicht der Komponenten der MDM-Plattform

Durch die MDM-Plattform werden folgende Kommunikations- und Anwendungsszenarien unterstützt:

- Interessenten, Datennehmer und Datengeber können über die Web-GUI mit dem Metadatenverzeichnis kommunizieren, um Dienste wie Recherchieren oder Registrieren in Anspruch zu nehmen. Um bestimmte Inhalte des Metadatenverzeichnisses einsehen oder ändern zu können, muss zuvor eine Authentisierung an der MDM-Plattform Security-Komponente durchlaufen werden.
- Datennehmersystem und Datengebersystem können nach Authentisierung über die Security-Komponente eine M2M-Kommunikation mit dem Brokersystems aufbauen, um Daten abzuliefern bzw. um Daten anzufordern.

3 Datenaustausch-Formate

Um Mobilitätsdaten zwischen Brokersystem sowie Datengeber- und Datennehmersystem austauschen zu können, werden folgende Datenformate vorgegeben:

- Die MDM-Plattform unterstützt das auf XML basierende Format DATEX II durch native Schnittstellen, um eine Nutzung der Plattform durch standard-konforme DATEX II-Implementierungen bei Datengebern oder Datennehmern zu ermöglichen.
- Um eine von konkreten Formaten unabhängige generische Schnittstelle zu schaffen, wird ein neues Datenformat zur Übermittlung bereitgestellt. Dabei handelt es sich um das sog. Containerformat, über das beliebige XML- und Binärdaten übertragen werden können.

Bei der Anlieferung eines Datenpakets an der Brokerschnittstelle des MDM wird die Validität der Daten überprüft und protokolliert. Dazu wird das Dateischema über die URL bezogen, die in der Publikationsbeschreibung hinterlegt ist. Für Publikationen im DATEX II-Format liegt es in der Verantwortung des Datengebers, das korrekte Dateischema zu hinterlegen. Für Publikationen im Containerformat wird das Standardschema bereits unter einer allgemein gültigen URL verfügbar gemacht. Bitte referenzieren Sie diese URL im „schemaLocation“-Attribut ihrer XML-Datenpakete, um Datennehmern eine automatische Validierung der Pakete mit den gleichen Voraussetzungen zu ermöglichen. Der MDM akzeptiert die Datenpakete unabhängig vom Validierungsergebnis und liefert diese auch an die Datennehmer aus, wenn das Ergebnis negativ ausfallen sollte.

3.1 DATEX II

DATEX II ist ein europaweiter Standard zum Austauschen von Mobilitätsdaten. Für diesen Abschnitt werden grundlegende Kenntnisse der DATEX II Spezifikation vorausgesetzt [DatexIISpec]. Für die MDM-Plattform wird die DATEX II Spezifikation in der Version 2.0 verwendet.

DATEX II definiert XML Strukturen für den Austausch von Mobilitätsdaten. Das zugrunde liegende Schema kann unter http://www.datex2.eu/ eingesehen werden. Die Nutzdaten sind anhand dieses Schemas zu definieren. DATEX II gibt nicht nur einen Standard für die Struktur der Nutzdaten vor, sondern regelt auch den Austauschprozess; dieser ist in Kapitel 4 genauer beschrieben.

Die dem DATEX II zugrunde liegenden Dokumente sind im Kapitel „Referenzierte Dokumente“ als [DatexIISpec] aufgeführt. Der Aufbau der DATEX II Nutzdaten ist für die MDM-Plattform nicht relevant, da diese die Daten unverändert weiterleitet und nicht auswertet.

DATEX II sieht nicht nur die Versendung kompletter Datenpakete vor, sondern auch eine Versendung von Änderungen zu vorherigen Versionen. Diese DATEX II Option wird von der MDM-Plattform **nicht** unterstützt: Sowohl Datengebersystem als auch MDM-Brokersystem müssen beim Versenden von Datenpaketen immer inhaltlich vollständige Daten verschicken.

Dies bedeutet, dass jedes Paket alle zum Zeitpunkt der Versendung des Pakets gültigen Datensätze der betreffenden Publikation enthält, die dem Datengeber bekannt sind. Es ist also nicht möglich, nur Veränderungen zum „letzten bekannten“ Stand zu versenden. Dies mag zunächst als Nachteil erscheinen, ist aber eine Anforderung, die zur Erhaltung der Skalierbarkeit des MDM-Systems unabdingbar ist. Der Nachteil einer teilweisen Redundanz wird hierfür billigend in Kauf genommen, da dieser durch die skalierbare Architektur der Plattform und die Leistungsfähigkeit moderner ITK-Infrastruktur aufgefangen wird. Es ist dabei zu berücksichtigen, dass die MDM-Plattform die Skalierbarkeitsbürde den Datengebern abnimmt.

3.2 Containerformat

Zusätzlich zu dem im vorherigen Abschnitt genannten DATEX II Standard wird durch die MDM-Plattform ein weiteres auf XML basierendes Modell zur Übermittlung von Daten unterstützt. Dieses Containerformat bezeichnete Datenformat wurde eigens für den Datenaustausch über den MDM geschaffen. Das Schema des Datenformats findet sich in der Containerformat Spezifikation [MCS]. Das Datenformat erlaubt es, neben den eigentlichen Nutzdaten, die in einem Body-Element enthalten sind, weitere Strukturinformationen im einem Header-Element zu übertragen, die insbesondere zur Steuerung des Kommunikationsprozesses benutzt werden.

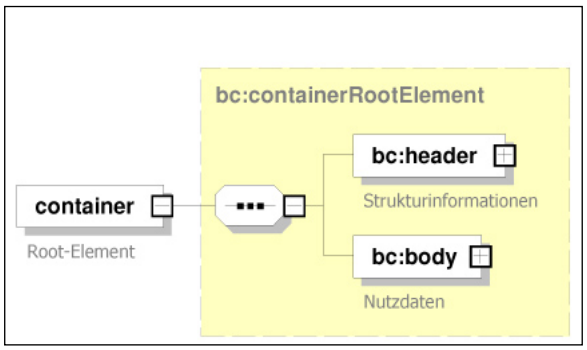

Abbildung 2: Übersicht Containerformat

Um das Modell flexibel zu halten, werden Format und Inhalt des Body-Elements nicht vorgegeben. So können nicht nur Daten im XML Format im Container transportiert werden, sondern auch Binärdaten.

4 Schnittstellen des MDM-Brokersystems

Das MDM-Brokersystem nimmt als Intermediär zwischen Datengeber-system und Datennehmersystem je nach Situation die Rolle des Clients oder die Rolle des Servers ein:

- Das Brokersystem kann als Client Daten vom Datengeber anfordern oder der Datengeber kann die Daten von sich aus an das Brokersystem schicken.
- Der Datennehmer kann seinerseits als Client Daten vom Brokersystem anfordern oder das Brokersystem kann die Daten von sich aus an den Datennehmer schicken.

Abbildung 3 zeigt die möglichen Wege auf, die zur Datenpaketübermittlung zwischen dem Datengeber und dem Brokersystem einerseits und dem Brokersystem und dem Datennehmer andererseits zur Verfügung stehen.

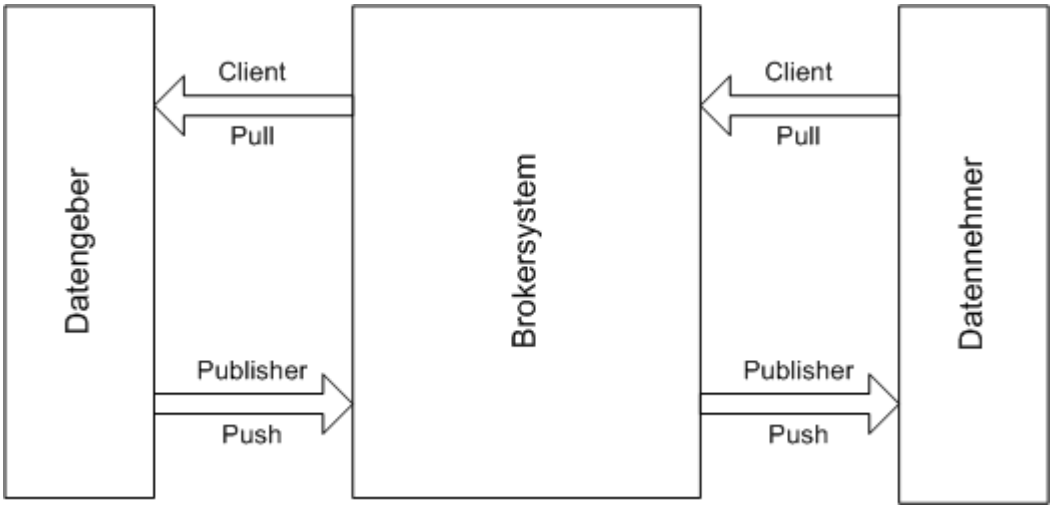

Abbildung 3: Schnittstellen zwischen Datengeber, Brokersystem und Datennehmer

Die Datenpakete, die vom Brokersystem empfangen oder gesendet werden, müssen im DATEX II Format oder im eigens definierten Containerformat vorliegen.

Als Übertragungsprotokolle für die jeweiligen Formate werden HTTPS und SOAP via HTTPS unterstützt. Für das Format DATEX II werden zudem das OTS 2- und das OCIT-C-Protokoll unterstützt.

Die folgende Tabelle zeigt, welche Kommunikationswege unterstützt werden. Dabei ist für jedes Datenformat (DATEX II / Container), Kommunikationsmuster (Client Pull / Publisher Push) und Protokoll (HTTPS, SOAP, OTS 2, OCIT-C), sofern unterstützt, das Kapitel vermerkt, in dem der entsprechende Kommunikationsweg beschrieben wird – unterschieden nach Datengeber- und Datennehmersystemen.

Zusätzlich ist jeweils angegeben, ob das Datengeber- bzw. Datennehmersystem gegenüber dem MDM als Client oder als Server auftritt. Client bedeutet hier, dass das System Anfragen an den MDM stellt bzw. aktiv die Verbindung zu diesem aufbaut.

Server bedeutet demgegenüber, dass das System vom MDM angesprochen wird und dessen Anfragen beantworten muss. In diesem Fall muss ein Netzwerkzugriff auf das anzubindende System von außen (durch den MDM) erlaubt sein.

		Datengebersystem				**Datennehmersystem**			
		HTTPS	SOAP	OTS2	OCIT	HTTPS	SOAP	OTS2	OCIT
DATEX II	Client Pull	4.3.1.1 Server	4.4.1.1 Server	-	-	4.3.2.1 Client	4.4.2.1 Client	-	4.6.3 Client
	Publisher Push	-	4.4.1.3 Client	4.5.4 Client	4.6.2 Client	-	4.4.2.3 Server	4.5.5 Client	-
Container	Client Pull	4.3.1.2 Server	4.4.1.2 Server	-	-	4.3.2.2 Client	4.4.2.2 Client	-	-
	Publisher Push	4.3.1.3 Client	4.4.1.4 Client	-	-	4.3.2.3 Server	4.4.2.4 Server	-	-

Tabelle 4: Übersicht über die Schnittstellen des MDM-Brokersystems

Kommt das SOAP-Verfahren zum Einsatz, gilt generell, dass am Publikations- bzw. Subskriptions-spezifischen Service Endpoint die WSDL des Brokerdienstes mit Hilfe des „?wsdl“-Requests abgefragt werden kann.

4.1 Unveränderlichkeitsversprechen

Die MDM-Plattform ist so konzipiert, dass sie die vom Datengeber angelieferten Daten unverändert an den oder die Datennehmer weiterreicht. Das Brokersystem darf den Nutzdatenanteil, die "Datex II Payload" der erhaltenen Datenpakete nicht verändern.

Für dieses Prinzip hat sich die Formulierung "Unveränderlichkeitsversprechen" etabliert.

Als eine Erweiterung dieses Prinzips kann angesehen werden, dass bei SOAP-Auslieferung auch der SOAP-Umschlag nicht geändert werden darf, wenn der Datengeber mit SOAP anliefert.

Eine wesentliche Auswirkung des "Unveränderlichkeitsversprechens" ist, dass sämtliche Namespace-Deklarationen, die sich auf die Datex II-Payload beziehen innerhalb des <d2LogicalModel>-Elements

definiert werden müssen, damit diese Deklarationen z. B. auch bei Anlieferung per SOAP und anschließender Weitergabe per HTTP Bestandteil der Payload bleiben (der SOAP-Envelope wird in diesen Fällen entfernt).

Hier ein Beispiel für eine funktionsfähige Implementierung unter Einhaltung des Unveränderlichkeitsverspechens:

```
<s:Body><d2LogicalModel
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="http://datex2.eu/schema/2/2_0
http://bast.s3.amazonaws.com/schema/1473413050792/DATEXIISchema_2_2_0.
xsd" modelBaseVersion="2"
xmlns="http://datex2.eu/schema/2/2_0"><exchange><supplierIdentificatio
n><country>de</country><nationalIdentifier>LMS-
HB</nationalIdentifier></supplierIdentification><target><address/><pro
tocol/></target><subscription><operatingMode>operatingMode0</operating
Mode>/headerInformation> ...
```

4.2 Verwendung der Schnittstellen

Bei Nutzung des HTTPS- oder SOAP-Protokolls gibt es drei unterschiedliche Operationsmodi für den Austausch der Daten, die alle von der MDM-Plattform unterstützt werden:

Modus	Beschreibung
Client Pull	Die Kommunikation wird vom Client (MDM-Brokersystem an Datengeber oder Datennehmersystem an MDM-Plattform) initiiert und die Daten werden als Response geschickt.
Publisher Push Periodic	Die Kommunikation wird vom Publisher (Datengebersystem an MDM-Plattform) in zeitlich vorgegebenen Intervallen initiiert.
Publisher Push on Occurrence	Die Kommunikation wird vom Publisher (Datengebersystem an MDM-Plattform oder MDM-Brokersystem an Datennehmer) immer dann initiiert, wenn sich die Daten ändern.

Tabelle 5: Operationsmodi des MDM

Das OTS 2-Protokoll [OTS2] arbeitet sitzungsbasiert und per Publish-Subscribe (Datenabonnement) und unterscheidet sich dadurch von den anderen angebotenen Schnittstellen des MDM.

Die gültige Spezifikation des OTS 2-Protokolls inklusive zugehöriger Schemadateien und WSDL kann über die OTS-Webseite bezogen

werden [OTS2]. Der dort verfügbare Stand entspricht inhaltlich der DIN SPEC (PAS) 91213 [OTS2DIN].

Zur Nutzung der OTS 2-Schnittstelle des MDM ist ein Client notwendig, der den OTS 2-Protokollstapel implementiert. Da OTS 2 aufgrund seiner umfangreichen Möglichkeiten (die für die MDM-Schnittstelle aber nur teilweise benötigt werden) eine höhere Komplexität als die anderen vom MDM angebotenen Schnittstellen hat, ist seine Verwendung abzuwägen.

Lohnenswert ist die Verwendung speziell dann, wenn im Client-System noch andere OTS 2-Schnittstellen betrieben werden oder geplant sind oder wenn der Vorteil des Push-Betriebs für Datennehmer-Clients genutzt wird. Dieser besteht darin, Daten des MDM im Push-Betrieb ohne Verzögerung durch zusätzliche Latenzzeiten bei Polling-Abfragen beziehen zu können, ohne dafür einen Server implementieren zu müssen, der eine Öffnung des eigenen Netzwerks für Zugriffe von außen (durch den MDM) notwendig macht (siehe Kapitel 4.5.5). Der Datennehmer wird also schnellstmöglich beliefert, muss aber dafür nicht sein Netzwerk nach außen öffnen wie bei den entsprechenden (Push) HTTPS- und SOAP-Protokolloptionen des MDM.

Als Datenmodell wird bei OTS 2 nur DATEX II verwendet wie im vorliegenden Dokument bzw. in [DatexIISpec] beschrieben. Bezüglich des Versands von kompletten Datenpaketen (im Gegensatz zum Versand von Änderungen) gelten die Ausführungen aus Kapitel 4.5.2. Eine Komprimierung findet nur für die eigentlichen Nutzdaten, nicht für den OTS 2-Protokollanteil statt (siehe Kapitel 4.5.3).

Das OCIT-Protokoll [OCIT-C] nutzt als Übertragungsverfahren SOAP auf Basis von http. Für die Implementierung wurde der OCIT-C-Standard in der Version 1.1_R1 vom 30.10.2014 angewendet. Für einen Datenaustausch mit dem MDM muss darüber hinaus eine Transportverschlüsselung mit TLS 1.0 oder höher und eine Authentifizierung mittels standardkonformer X.509v3-Zertifikate verwendet werden. Eine Authentifizierung mittels Username und Passwort wird MDM-seitig nicht unterstützt. Die entsprechenden Attribute des OCIT-C-Protokolls werden ignoriert.

Der OCIT-C-Funktionsumfang wird durch den MDM nur eingeschränkt und unter der Vorgabe einer spezifischen Nutzung von Protokollelementen angeboten. Als Datenmodell wird bei OCIT-C ähnlich wie bei OTS 2 nur DATEX II verwendet wie im vorliegenden Dokument bzw. in [DatexIISpec] beschrieben. Die OCIT-C Datenmodelle werden nicht unterstützt.

4.2.1 Datengeberseite

Gegenüber der Datengeberseite (dem Publisher) tritt das MDM-Brokersystem als Subscriber auf, der die Datenpakete entgegen nimmt. Das Brokersystem kann je nach Verfahren die Rolle eines Servers oder Clients einnehmen.

Bei Nutzung des OTS 2-Protokolls agiert das Brokersystem in der Rolle eines OTS 2-Distributor, das Datengebersystem in der Rolle eines OTS 2-Publisher.

Bei Nutzung des OCIT-C-Protokolls agiert das Brokersystem als Server und das Datengebersystem als Client.

4.2.2 Datennehmerseite

Gegenüber der Datennehmerseite (dem Subscriber) tritt das MDM-Brokersystem als Publisher auf, der die Datenpakete bereithält. Das Brokersystem kann je nach Verfahren die Rolle eines Servers oder Clients einnehmen.

Bei Nutzung des OTS 2-Protokolls agiert das Brokersystem in der Rolle eines OTS 2-Distributor, das Datennehmersystem in der Rolle eines OTS 2-Subscriber.

Bei Nutzung des OCIT-C-Protokolls agiert das Brokersystem als Server und das Datennehmersystem als Client.

4.3 HTTPS-Schnittstelle

4.3.1 Datengeberseite

4.3.1.1 Client Pull HTTPS (DATEX II)

Beim Client Pull Austauschverfahren fordert das MDM-Brokersystem zyklisch das Datengebersystem auf, seine Daten an der MDM-Plattform abzuliefern. Das verwendete Zeitintervall muss bei der Konfiguration des Datenangebots im Metadatenverzeichnis konfiguriert werden. Für diesen Austausch gelten aus dem Simple HTTP Server Profile des [DatexIIPSM] die Punkte C1–C12.

Dabei ist zu berücksichtigen, dass die weiteren, optionalen Regeln keine Anwendung finden. Die Optionen zur Authentisierung (C13, C14, C17) finden keine Anwendung, da sie bei der Verwendung des für den MDM verpflichtenden HTTPS-Verfahrens obsolet sind. C18-C27 entfallen, da die Optionen sich nur auf die optionale Bereitstellung von DATEX II-Daten in Dateiform beziehen, die beim MDM nicht zur Anwendung kommt.

4.3.1.1.1 Request an den Datengeber

Das MDM-Brokersystem schickt einen HTTPS GET-Request zum Datengebersystem, von dem die Daten abgeholt werden sollen. Die MDM-Plattform ist in der Lage, Datengebersysteme, die ein Pull-Verfahren abonniert haben, zu identifizieren und in definierten Abständen Requests an diese zu schicken.

Über die MDM-Administrations-Komponente muss der Datengeber die Publikations-spezifische Server-URL in der Publikations-Konfiguration hinterlegen.

Das Brokersystem versendet die Requests mit einem „If-Modified-Since“ Header-Field immer dann, wenn das Datengebersystem in seiner Response das Header-Field „Last-Modified“ gesetzt hatte (vgl. [HTTP/1.1]). Das Datengebersystem sollte dieses Header-Field immer setzen, um die MDM-Plattform in die Lage zu versetzen, dieses Feature anwenden zu können. Hierdurch wird die Übertragung bereits abgeholter Datenpakete vermieden.

Beispiel:

Wenn die Response des vorhergehenden Datenpakets folgende Headerzeile enthält

```
Last-Modified: Sat, 29 Oct 1994 19:43:31 GMT
```

wird das nächste Datenpaket mit einem Request angefordert, der folgende Header-Zeile enthält:

```
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

4.3.1.1.2 Response an die MDM-Plattform

Das Datengebersystem muss nach Erhalt des Requests eine HTTPS Response erzeugen, deren Message-Body aus den angeforderten DATEX II Daten besteht. Gemäß [DatexIIPSM] Abschnitt 4 hat die Response den Content–Type „text/xml; charset=utf-8“ und sollte als GZIP-Encoding vorliegen.

Das MDM-Brokersystem nimmt diese Daten an und speichert sie in einem Paketpuffer. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

4.3.1.2 Client Pull HTTPS (Container)

Das MDM-Brokersystem fordert das Datengebersystem zyklisch auf, ein Datenpaket zu einer Publikation an der MDM-Plattform abzuliefern. Das verwendete Zeitintervall muss bei der Konfiguration des Datenangebots im Metadatenverzeichnis konfiguriert werden.

4.3.1.2.1 Request an den Datengeber

Das Brokersystem schickt einen HTTPS GET-Request zum Datengebersystem. Als Parameter wird dabei die Publikations-ID übergeben, zu der ein Datenpaket geliefert werden soll.

Über die MDM-Administrations-Komponente muss der Datengeber seine URL in der Publikations-Konfiguration hinterlegen.

Die URL des Datengebersystems aus der Publikationskonfiguration wird durch Anhängen der Publikations-ID ergänzt:

Beispiel:

Datengeber konfiguriert im MDV als URL zur Abholung:

```
https://<DG-Maschine>/<context>
```

Die ID der zugehörigen Publikation sei 2000002. Daraus ergibt sich folgende URL für den Aufruf durch das MDM-Brokersystem:

```
https://<DG-Maschine>/<context>?publicationID=2000002
```

4.3.1.2.2 Response an die MDM-Plattform

Auf den Request muss das Datengebersystem mit einer HTTPS-Response antworten. Der Content-Type der Response muss vom Typ „text/xml“ sein und sollte als GZIP-Encoding vorliegen. Auch nicht komprimierter Inhalt kann von der MDM-Plattform verarbeitet werden. Der Message-Body muss aus dem angeforderten Datenpaket bestehen. Als Statuscodes sind die Standard HTTP Statuscodes [HTTP/1.1] zu verwenden, wobei die in Tabelle 6 beschriebenen Bedeutungen gelten.

Beschreibung	
Request	GET /anfrageServlet?publicationID=2000002 HTTP/1.1 Host: Datengeberhost Accept-Encoding: GZIP
Response	HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <container> ... </container>
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 400: Es wurde kein Publikationsparameter übergeben - 404: Publikationsparameter konnte nicht zugeordnet werden

Tabelle 6: Request/Response zwischen Datengebersystem/MDM-Plattform beim Client Pull HTTPS

4.3.1.3 Publisher Push HTTPS (Container)

Das Datengebersystem muss ein Datenpaket zu einer Publikation an das MDM-Brokersystem schicken.

4.3.1.3.1 Request an das MDM-Brokersystem

Das Datengebersystem muss einen HTTPS POST-Request mit einer Nachricht im Containerformat zum MDM-Brokersystem schicken. Dabei müssen die Publikations-ID im Header-Element und die Nutzdaten im Body-Element der Containernachricht übergeben werden.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-Interface/srv/container/v1.0
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-Interface/srv/container/v1.0
```

4.3.1.3.2 Response an den Datengeber

Als Antwort auf den Request erhält das Datengebersystem eine HTTPS Response. Der Message-Body ist leer, als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 7 beschriebenen Bedeutungen gelten.

Beschreibung	
Request	Request POST /datenabgabe HTTP/1.1 Host: mdmhost Content-Type : text/xml Accept-Encoding: GZIP <container> ... </container>
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 400: Es wurde kein Publikationsparameter oder keine Daten übergeben - 404: Publikationsparameter konnte nicht zugeordnet werden oder die Publikation ist nicht mehr gültig

Tabelle 7: Request/Response zwischen Datengebersystem/MDM-Plattform beim Publisher Push HTTPS

4.3.2 Datennehmerseite

4.3.2.1 Client Pull HTTPS (DATEX II)

Beim Client Pull Austauschverfahren muss das Datennehmersystem das MDM-Brokersystem auffordern, die Daten zu übermitteln.

4.3.2.1.1 Request an die MDM-Plattform

Das Datennehmersystem muss einen HTTPS GET-Request an die URL der MDM-Plattform schicken. Aufgrund der Subskriptions-ID ist der zugehörige Paketpuffer sowie das Datenpaket festgelegt.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-
Interface/srv/<Subskriptions-
ID>/clientPullService?subscriptionID=<Subskriptions-ID>
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000000/clientPullService?subscriptionID=2000000
```

Das Brokersystem unterstützt Requests, die ein „If-Modified-Since“ Header-Field aufweisen. Zu diesem Zweck enthalten die Responses des Brokersystems immer das Header-Field „Last-Modified“ (vgl. [HTTP/1.1]). Falls das Datennehmersystem dieses Feature nutzen möchte, muss es immer den Wert aus dem letzten Last-Modified

Header-Field mitsenden. Hierdurch wird die Übertragung bereits abgeholter Datenpakete vermieden. Es wird dringend empfohlen, dieses Feature auf Datennehmerseite zu implementieren.

Beispiel:

Wenn die Response des vorhergehenden Datenpakets z.B folgendes Header-Field enthält

```
Last-Modified: Sat, 29 Oct 1994 19:43:31 GMT
```

muss das nächste Datenpaket mit einem Request angefordert werden, der folgendes Header-Field enthält:

```
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

4.3.2.1.2 Response an den Datennehmer

Das MDM-Brokersystem erzeugt nach Erhalt des Requests eine HTTPS Response. Dazu werden aufgrund der Subskriptions-ID der zugehörige Paketpuffer sowie das passende Datenpaket ermittelt. Der Inhalt des Datenpakets wird im Body der Response an den Datennehmer übermittelt. Gemäß DATEX II Client Pull HTTP Profil [DatexIIPSM] Abschnitt 4 hat die Response den Content Type „text/xml; charset=utf-8“.

4.3.2.2 Client Pull HTTPS (Container)

Beim Client Pull Austauschverfahren muss das Datennehmersystem das MDM-Brokersystem auffordern, die Daten zu übermitteln. Um welche Subskription es sich dabei handelt, muss durch einen Request-Parameter spezifiziert werden.

4.3.2.2.1 Request an die MDM-Plattform

Das Datennehmersystem muss einen HTTPS GET-Request an die MDM-Plattform schicken. Als Parameter muss die Subskriptions-ID übermittelt werden, zu der ein Datenpaket geliefert werden soll.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-
Interface/srv/container/v1.0?subscriptionID=<Subskriptions-ID>
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/container/v1.0?subscriptionID=2000000
```

4.3.2.2.2 Response an das Datennehmersystem

Das MDM-Brokersystem erzeugt nach Erhalt des Requests eine HTTPS Response. Als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 8 beschriebenen Bedeutungen gelten. Der Content-Type der Response ist vom Typ „text/xml“ und wird GZIP-komprimiert versendet. Der Message-Body der Response besteht aus dem angeforderten Datenpaket.

Beschreibung	
Request	Request GET /BASt-MDM-Interface/srv/container/v1.0?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <container> ... </container>
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 204: Kein Datenpaket im Paketpuffer zur Subskription - 400: Kein Subskriptionsparameter - 404: Keine oder eine nicht mehr gültige Subskription zum Subskriptionsparameter gefunden

Tabelle 8: Request/Response zwischen MDM-Plattform/Datennehmersystem beim Client Pull HTTPS

4.3.2.3 Publisher Push HTTPS (Container)

Das MDM-Brokersystem schickt ein Datenpaket zu einer Subskription an ein Datennehmersystem.

4.3.2.3.1 Request an das Datennehmersystem

Das MDM-Brokersystem schickt einen HTTPS POST-Request zum Datennehmersystem, in dem die Subskriptions-ID im Header-Element sowie die Nutzdaten im Body-Element der Containernachricht übergeben werden.

Über die MDM-Administrations-Komponente muss der Datennehmer seine URL in der Subskriptions-Konfiguration hinterlegen.

4.3.2.3.2 Response an die MDM-Plattform

Auf den Request muss das Datennehmersystem mit einer HTTPS Response antworten.

Der Message-Body soll leer sein, als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 9 beschriebenen Bedeutungen gelten.

	Beschreibung
Request	Request POST /datenabgabe HTTP/1.1 Host: datennehmerhost Content-Type : text/xml Accept-Encoding: GZIP <container> ... </container>
Response	Response HTTP/1.1 200 OK
Statuscodes	Statuscodes Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 400: Es wurde kein Subskriptionsparameter oder keine Daten übergeben - 404: Subskriptionsparameter konnte nicht zugeordnet werden

Tabelle 9: Request/Response zwischen MDM-Brokersystem/Datennehmersystem beim Publisher Push HTTPS

4.4 SOAP-Schnittstelle

4.4.1 Datengeberseite

4.4.1.1 Client Pull SOAP (DATEX II)

Beim Client Pull SOAP Austauschverfahren fordert das MDM-Brokersystem das Datengebersystem auf, seine Daten an der MDM-Plattform abzuliefern.

4.4.1.1.1 Anbieten eines Webservices

Das Datengebersystem muss einen Webservice anbieten, der aufgrund der DATEX II Pull WSDL [DatexIIPull] definiert ist. Als Input wird dabei nichts erwartet, als Output bekommt das MDM-Brokersystem die angeforderten Daten im DATEX II Format zurück.

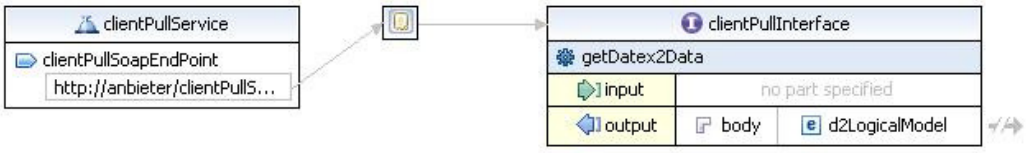

Abbildung 4: Webservice Datengebersystem/MDM-Brokersystem: DATEX II Client Pull

Über die MDM-Administrations-Komponente muss der Datengeber seine URL in der Publikations-Konfiguration hinterlegen.

4.4.1.1.2 Aufrufen eines Webservices

Das MDM-Brokersystem stellt einen aufgrund der DATEX II Pull WSDL [DatexIIPull] definierten Webservice-Client zum Aufruf von Webservices bereit. Dieser Webservice muss Daten gemäß des Schemas [DatexIISchema] zurückliefern.

Das Brokersystem identifiziert die Datengebersysteme, die ein Pull-Verfahren abonniert haben, sowie die zugehörigen Service-Endpoints im Metadatenverzeichnis und ruft diese zyklisch gemäß der konfigurierten Publikationsfrequenz auf. Die nach dem Aufruf empfangenen Daten werden für die Abgabe an potentielle Datennehmer in entsprechenden Paketpuffern zwischengespeichert. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

4.4.1.2 Client Pull SOAP (Container)

Beim Client Pull SOAP Austauschverfahren fordert das MDM-Brokersystem das Datengebersystem zyklisch auf, seine Daten an der MDM-Plattform abzuliefern. Das verwendete Zeitintervall muss bei der Konfiguration des Datenangebots im Metadatenverzeichnis konfiguriert werden.

4.4.1.2.1 Anbieten eines Webservices

Das Datengebersystem muss einen Webservice anbieten, der als Input die Parameter Publikations-ID und Zeitstempel mit einem Erstellungsdatum gemäß den Elementen des Container-Modell-Schemas erwartet. Das Datengebersystem muss zu der übergebenen Publikations-ID ein Datenpaket im Containerformat erzeugen und zurückschicken.

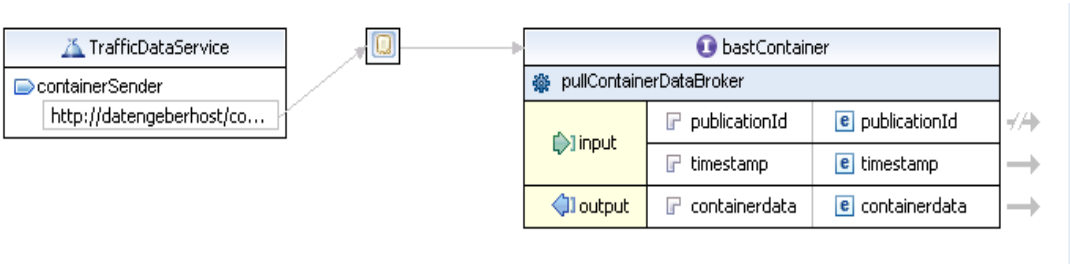

Abbildung 5: Webservice Datengebersystem/MDM-Brokersystem: Container Client Pull

Über die MDM-Administrations-Komponente muss der Datengeber den Service-Endpoint im URL-Attribut der Publikations-Konfiguration hinterlegen.

4.4.1.2.2 Aufrufen eines Webservices

Das MDM-Brokersystem stellt einen gemäß Containerformat Spezifikation [MCS] definierten Webservice-Client zum Aufruf von Webservices bereit.

Das Brokersystem identifiziert die Datengebersysteme, die ein Pull-Verfahren abonniert haben, sowie die zugehörigen Service-Endpoints im Metadatenverzeichnis und ruft diese zyklisch gemäß der konfigurierten Publikationsfrequenz auf. Die nach dem Aufruf empfangenen Daten werden für die Abgabe an potentielle Datennehmer in entsprechenden Paketpuffern zwischengespeichert. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

4.4.1.3 Publisher Push SOAP (DATEX II)

Beim Publisher Push Austauschverfahren muss das Datengebersystem von sich aus die Daten an die MDM-Plattform anliefern. Dabei muss eine entsprechende SOAP-Schnittstelle verwendet werden. Ob die Daten aufgrund eines Ereignisses (on occurrence) oder periodisch (periodic) erzeugt und zur MDM-Plattform geliefert werden, ist für die Funktionsweise des MDM-Bokersystems unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

4.4.1.3.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der aufgrund der Spezifikation DATEX II Push WSDL [DatexIIPush] definiert ist. Als Input werden die zu überliefernden Daten erwartet, als Output bekommt das Datengebersystem Bestätigungsdaten im DATEX II Format zurück.

Der Output besteht aus einer Bestätigung des Empfangs.

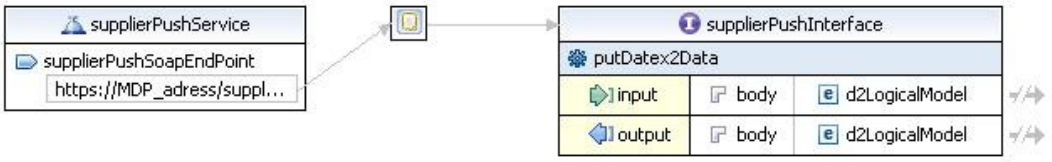

Abbildung 6: Webservice Datengebersystem/MDM-Brokersystem: DATEX II Publisher Push

In der URL des Service-Endpoints am Brokersystem wird die ID der Publikation eingetragen, in die die Datenpakete eingestellt werden sollen.

Die URL ist folgendermaßen aufgebaut:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-Interface/srv/<publication
ID>/supplierPushService
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000002/supplierPushService
```

4.4.1.3.2 Aufrufen des Webservices

Das Datengebersystem muss einen aufgrund der DATEX II Push WSDL [DatexIIPush] definierten Webservice Client zum Aufruf des Webservices bereitstellen. Der Webservice muss am Publikations-spezifischen Service-Endpoint des MDM-Brokersystems die Daten anliefern. Das MDM-Brokersystem nimmt diese Daten an und speichert sie in einem Paketpuffer. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

4.4.1.4 Publisher Push SOAP (Container)

Beim Publisher Push Austauschverfahren muss das Datengebersystem von sich aus die Daten an die MDM-Plattform anliefern. Dabei muss eine entsprechende SOAP-Schnittstelle verwendet werden. Ob die Daten aufgrund eines Ereignisses (on occurrence) oder periodisch (periodic) erzeugt und zur MDM-Plattform geliefert werden, ist für die

Funktionsweise des MDM-Bokersystems unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

4.4.1.4.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der als Input die Datenstruktur des Containerformats gefüllt mit der Publikations-ID im header-Element und einem Datenpaket im body-Element erwartet und als Output eine Statusnachricht zurückliefert.

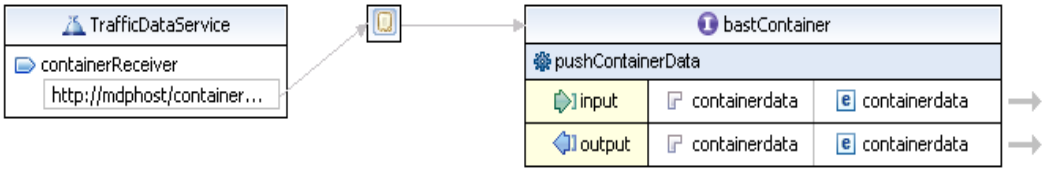

Abbildung 7: Webservice Datengebersystem/MDM-Brokersystem: Container Publisher Push

4.4.1.4.2 Aufrufen des Webservices

Das Datengebersystem muss einen Webservice Client gemäß Containerformat Spezifikation [MCS] zum Aufruf des Webservices bereitstellen.

Der SOAP-Endpoint des Brokersystems lautet:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-Interface/srv/container/v1.0
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-Interface/srv/container/v1.0
```

4.4.2 Datennehmerseite

4.4.2.1 Client Pull SOAP (DATEX II)

Beim Client Pull SOAP Austauschverfahren muss das Datennehmersystem die MDM-Plattform auffordern, Daten an das Datennehmersystem zu schicken.

4.4.2.1.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der aufgrund der Spezifikation [DatexIIPull] definiert ist. Als Input wird dabei in der URL die Subskriptions-ID erwartet, als Output bekommt der Datennehmer die angeforderten Daten im DATEX II Format zurück. Aufgrund der übermittelten Subskriptions-ID kann die MDM-Plattform den zugehörigen Paketpuffer sowie das Datenpaket ermitteln.

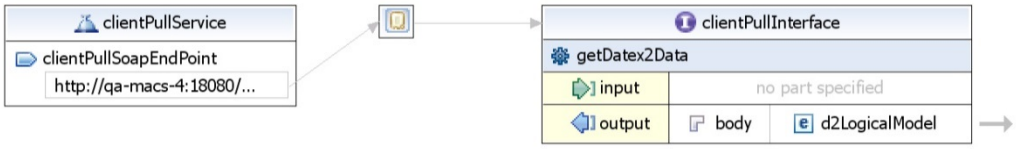

Abbildung 8: Webservice MDM-Brokersystem/Datennehmersystem: DATEX II Client Pull

4.4.2.1.2 Aufrufen des Webservices

Das Datennehmersystem muss einen aufgrund der Spezifikation [DatexIIPull] definierten Webservice-Client zum Aufruf des Webservices bereitstellen. Als Input-Parameter muss die entsprechende Subskriptions-ID in der URL mitgeführt werden.

Der SOAP-Endpoint des Brokersystems lautet:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-
Interface/srv/<Subskriptions-ID>/clientPullService
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000000/clientPullService
```

4.4.2.2 Client Pull SOAP (Container)

Beim Client Pull SOAP Austauschverfahren muss das Datennehmersystem die MDM-Plattform auffordern, Daten an das Datennehmersystem zu schicken.

4.4.2.2.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der als Input eine Subskriptions-ID und einen Zeitstempel (beinhaltet den Erstellungszeitpunkt der Anfrage) erwartet. Als Output werden die Daten im Containerformat zurückgeliefert.

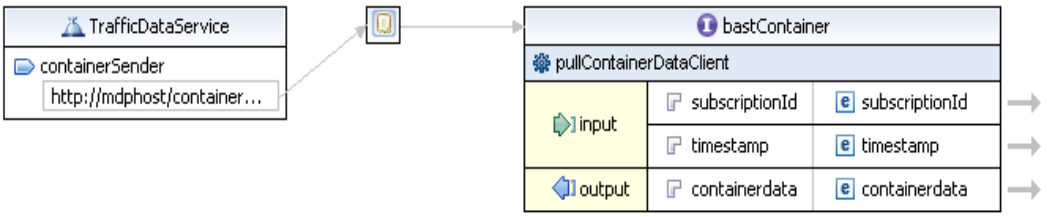

Abbildung 9: Webservice MDM-Brokersystem/Datennehmersystem: Container Client Pull

4.4.2.2.2 Aufrufen des Webservices

Das Datennehmersystem muss einen Webservice-Client gemäß Containerformat Spezifikation [MCS] zum Aufruf des Webservices bereitstellen.

Der SOAP-Endpoint des Brokersystems lautet:

```
https://<BASt-MDM-Broker-Server>/BASt-MDM-Interface/srv/container/v1.0
```

Beispiel:

```
https://broker.mdm-portal.de/BASt-MDM-Interface/srv/container/v1.0
```

4.4.2.3 Publisher Push SOAP (DATEX II)

Beim Publisher Push Austauschverfahren liefert das MDM-Brokersystem von sich aus die Daten an die Datennehmersysteme. Dabei wird eine entsprechende SOAP-Schnittstelle verwendet. Ob die Daten aufgrund eines Ereignisses (on occurrence) oder periodisch (periodic) erzeugt und beim MDM angeliefert werden, ist dabei unerheblich, der Mechanismus zur Abgabe an den Datennehmer ist identisch.

4.4.2.3.1 Anbieten eines Webservices

Das Datennehmersystem muss einen Webservice anbieten, der aufgrund der Spezifikation [DatexIIPush] definiert ist. Als Input werden die angeforderten Daten erwartet, als Output bekommt die MDM-Plattform Bestätigungsdaten im DATEX II Format zurück. Das Format des Input-Parameters entspricht dem DATEX II Schema [DatexIISchema].

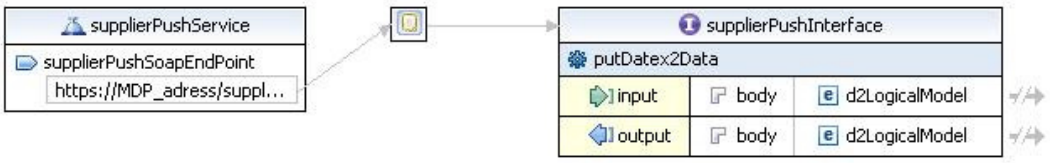

Abbildung 10: Webservice MDM-Brokersystem/Datennehmersystem: DATEX II Publisher Push

4.4.2.3.2 Aufrufen des Webservices

Das MDM-Brokersystem stellt einen auf Basis von [DatexIIPush] definierten Webservice-Client zum Aufruf der Datennehmer-Webservices bereit. Über die MDM-Administrations-Komponente muss der Datennehmer seinen Service-Endpoint in der Subskriptions-Konfiguration hinterlegen.

Das Brokersystem identifiziert diese Datennehmersysteme und startet einen entsprechenden Webservice-Aufruf.

Konnte die Übertragung der Daten erfolgreich abgeschlossen werden, erwartet das Brokersystem vom Datennehmersystem eine entsprechende Bestätigungsnachricht:

```
<D2LogicalModel:d2LogicalModel modelBaseVersion="2"
xsi:schemaLocation="http://datex2.eu/schema/2/2_0/
DATEXIISchema_2_2_0.xsd"
xmlns:D2LogicalModel="http://datex2.eu/schema/2/2_0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<D2LogicalModel:exchange>
        <D2LogicalModel:response>acknowledge</D2LogicalModel:response>
 </D2LogicalModel:exchange>
...
</D2LogicalModel:d2LogicalModel>
```

4.4.2.4 Publisher Push SOAP (Container)

Beim Publisher Push Austauschverfahren liefert das MDM-Brokersystem von sich aus die Daten an die Datennehmersysteme. Dabei wird eine entsprechende SOAP-Schnittstelle verwendet. Ob die Daten aufgrund eines Ereignisses (on occurrence) oder periodisch (periodic) erzeugt und beim MDM angeliefert werden, ist dabei unerheblich, der Mechanismus zur Abgabe an den Datennehmer ist identisch.

4.4.2.4.1 Anbieten eines Webservices

Das Datennehmersystem muss einen Webservice anbieten, der aufgrund der Containerformat Spezifikation [MCS] definiert ist. Als

Input muss ein Datenpaket vom Typ des Containerformates akzeptiert werden und als Output ist eine Statusnachricht zu liefern.

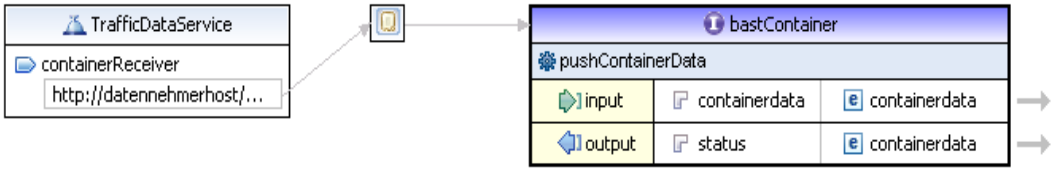

Abbildung 11: Webservice MDM-Brokersystem/Datennehmersystem: Container Publisher Push

4.4.2.4.2 Aufrufen des Webservices

Das MDM-Brokersystem stellt einen auf Basis der Containerformat Spezifikation [MCS] definierten Webservice-Client zum Aufruf der Datennehmer-Webservices bereit. Über die MDM-Administrations-Komponente muss der Datennehmer seinen Service-Endpoint im URL-Attribut der Subskriptions-Konfiguration hinterlegen.

Das Brokersystem identifiziert diese Datennehmersysteme und startet einen entsprechenden Webservice-Aufruf.

Konnte die Übertragung der Daten erfolgreich abgeschlossen werden, erwartet das Brokersystem vom Datennehmersystem eine entsprechende Statusnachricht.

4.5 OTS 2-Schnittstelle

4.5.1 Ablauf

Zum Liefern oder Abholen von Daten muss der externe Client zunächst eine Sitzung mit dem OTS 2-Server im MDM aufbauen.

Daraufhin findet innerhalb der Sitzung eine Datenbestellung (Abonnement) durch den vorgesehenen Empfänger (Datennehmer oder MDM) statt. Der Datensender (MDM oder Datengeber) versendet daraufhin ebenfalls innerhalb der bestehenden Sitzung automatisch und fortlaufend (ohne weitere Abfragen) die gewünschten Daten solange, bis die Bestellung beendet wird.

4.5.2 Funktionsumfang

Die für die Kommunikation mit dem MDM verwendeten Methoden der OTS 2 Aktivitätsschicht sind ATie, ASubscribe (für Datennehmer) / ASnippet (für Datengeber), AUnsubscribe und AUntie. Die genutzten Methodenaufrufe sind onATied, onASnippet (für Datennehmer) / onASubscribe (für Datengeber), onAUnsubscribe und onAUntied. Es werden die Dateninhalte application, subscriptionAny und dataAny transportiert.

Es wird also nur der OTS 2-Protokollumfang benötigt, der die Übermittlung beliebiger zeitaktueller DATEX II-Pakete per subscriptionAny ermöglicht. Singuläre Anfragen (Queries) und Befehle (Commands) kommen nicht zum Einsatz. Historische Daten können vom MDM nicht bezogen werden. Auch spezielle Abfragen zum Auslesen des aktuellen Zustands und dann darauf folgender Änderungen werden nicht unterstützt. Datenpakete von Datengebern müssen daher immer den vollständigen aktuellen Zustand beschreiben.

Datennehmer erhalten keine Möglichkeit, eine Auswahl von Objekten eines Services abzufragen. Datenpakete werden immer vollständig abgegeben (MDM-Subskription). Ggf. sollten Datengeber ihr Datenangebot deshalb in mehrere Publikationen aufteilen, um Datennehmer in die Lage zu versetzen, nur eine Teilmenge aus dem verfügbaren Datenangebot zu abonnieren.

4.5.3 DATEX II-Komprimierung für OTS 2

Auf Datennehmerseite liefert der MDM die Daten grundsätzlich komprimiert aus. Im Gegensatz zur Nutzung der Protokolle HTTPS und SOAP wird bei OTS 2 allerdings nur die DATEX II Nutzlast komprimiert und nicht das vollständige OTS 2 Paket. Betrachtet man das Paket auf HTTP-Ebene, so handelt es sich demzufolge um ein unkomprimiertes Paket. Aufgrund der Anforderungen an die Durchlaufzeit von Datenpaketen durch den MDM und der relativ langen Rechenzeit, die für die Komprimierung eines einzelnen Datenpakets benötigt wird, wurde entschieden, nur die DATEX II-Payload komprimiert zu

übertragen und den subskriptionsspezifischen individuellen OTS 2-Rahmen unkomprimiert zu übertragen. So kann der MDM auch bei Abgabe über OTS 2 ein einmal komprimiertes Datenpaket an viele Datennehmer ausliefern.

Ein OTS 2-Snippet vom Typ acDataAnyType (andere Typen werden nicht verwendet) nimmt zu diesem Zweck ein BASE64Binary-kodiertes Binärpaket auf, welches das DATEX II-Paket in GZIP-komprimierter Form enthält.

Das Binärpaket wird dazu in ein <binary>-Element eingebettet. Das Attribut type des <binary>-Elements kennzeichnet die Art der übertragenen Daten und wird hier mit „base64BinaryDatex2Gzip“ belegt.

Daraus ergibt sich folgender Aufbau innerhalb eines OTS 2-Snippets:

```
<dataAny>
   <binary type="base64BinaryDatex2Gzip">
      PGQyTG9naWNhbE1vZGVsIHhtbG5zPSJodHRwOi8vZGF0ZXgyLmV1L3NjaGVt...
   </binary>
</dataAny>
```

Zur Wiederherstellung des originalen DATEX II-Pakets muss ein Datennehmer den Inhalt des <binary>-Elements aus dem dataAny-Snippet zunächst BASE64Binary-dekodieren und anschließend GZIP-dekomprimieren.

OTS 2-Datengeber können neben der klassischen Komprimierung auf HTTP-Ebene diese Variante der Komprimierung bei der Anlieferung ebenfalls nutzen; die Nutzung dieses Verfahrens wird empfohlen.

4.5.4 OTS 2 Publish

Der Datengeber nimmt bei Verwendung des OTS 2-Protokolls die Rolle eines OTS 2-Publisher ein.

Je MDM-Publikation muss eine separate Verbindung zu dem dafür vorgesehenen Service-Endpoint aufgebaut werden (OTS 2-Methode ATie).

Die URL des Service-Endpoints ist folgendermaßen aufgebaut:

```
soap.tls://<BASt-MDM-Broker-Server>/BASt-MDM-OTS2-
Interface/pub/<publicationID>
```

Beispiel:

```
soap.tls://broker.mdm-portal.de/BASt-MDM-OTS2-Interface/pub/2004000
```

Die Verbindungen werden normalerweise dauerhaft – so lange wie möglich – gehalten (und nicht z.B. jede Minute neu aufgebaut). Die Authentisierung findet nur beim Verbindungsaufbau statt (s.u.).

Die nachfolgende Abbildung 12 zeigt einen beispielhaften Ablauf mit Verbindungsaufbau, Bestellung durch den MDM, Datenlieferung durch den Client (hier sind nur zwei Lieferungen eingezeichnet) und Verbindungsabbau durch den Client (ein Verbindungsabbau in umgekehrter Richtung durch den MDM wäre auch möglich). Der Datengeber (Client) ist auf der linken Seite eingezeichnet.

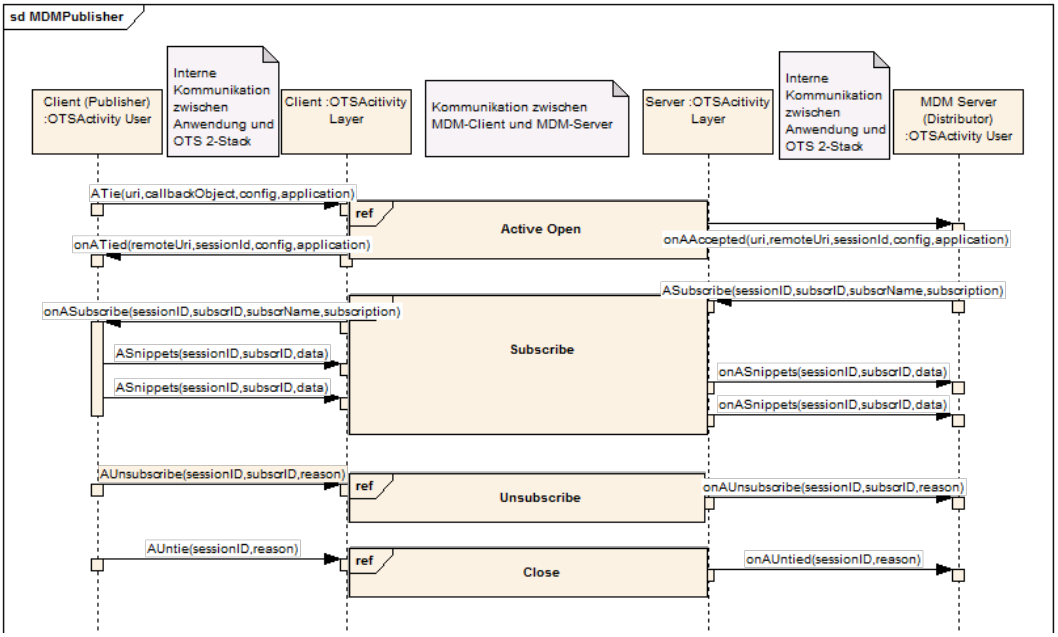

Abbildung 12: Sequenzdiagramm OTS 2-Kommunikation zwischen Datengeber und MDM

4.5.4.1 Verbindungsaufbau

Das Datengebersystem muss unter Verwendung seines Maschinenzertifikats eine TLS-Verbindung in Richtung MDM mit der OTS 2-Protokollbindung „SOAP/HTTP mit TLS Verschlüsselung“ aufbauen (OTS 2-Methode ATie, im URL Schema-Feld steht „soap.tls:“).

Der Datengeber muss in den Konfigurationsinformationen folgende spezielle Features setzen:

- „a_publisher=1“ (gibt an, dass es sich um einen Datengeber handelt)

- „a_c_datex_any_mdm=1“ (codiert die speziellen Randbedingungen für den Einsatz des OTS 2-Protokolls im MDM)
- „t_targetURI“ mit der vollständigen Ziel-URI, zum Beispiel „t_targetURI=soap.tls://broker.mdm-portal.de/BASt-MDM-OTS2-Interface/pub/2004000“ (wird vom MDM-Server OTS 2 intern benötigt)

Zum Verbindungsaufbau siehe auch [OTS2] Kapitel 7.6.4.1 und 7.6.4.2.

4.5.4.2 Bestellung

Der Datengeber muss gemäß OTS 2-Protokoll nach dem erfolgten Verbindungsaufbau (OTS 2-Methodenaufruf onATied) warten, bis der MDM seine Bestellung aufgibt (OTS 2-Methodenaufruf onASubscribe).

Die Bestellung ist vom Typ acSubscriptionAnyType. Die zu liefernden Daten liegen bereits durch die Auswahl des Service-Endpoints beim MDM fest und werden deshalb in der Bestellung nicht näher spezifiziert.

Zur Bestellung siehe auch [OTS2] Kapitel 7.6.7.1 und 7.6.7.2.

4.5.4.3 Datenlieferung

Der Datengeber liefert nach Empfang der Bestellung regelmäßig seine Daten aus (OTS 2-Methode ASnippet). Der Datengeber ist dafür verantwortlich, dem MDM seine Datenpakete in der vereinbarten Anlieferungsfrequenz (gemäß Konfiguration des Datenangebots im Metadatenverzeichnis) zu übermitteln.

Die Datenpakete müssen den Typ acDataAnyType benutzen. Die Daten darin werden als DATEX II-Paket oder (empfohlen) als BASE64-encodiertes und GZIP-komprimiertes DATEX II-Binärpaket (siehe Kapitel 4.5.3) erwartet.

Zur Datenlieferung siehe auch [OTS2] Kapitel 7.6.7.3 und 7.6.7.4.

4.5.4.4 Verbindungsabbau

Eine bestehende Verbindung kann von beiden Seiten wieder abgebaut werden (OTS 2-Methoden AUnSubscribe zum Beenden der Bestellung bzw. Datenauslieferung und danach AUntie zum Verbindungsabbau).

Vom MDM werden Verbindungen nur abgebaut, wenn eine Publikation oder Subskription aus der Verwaltung (Metadatenverzeichnis) heraus in einen inaktiven Zustand versetzt wird oder wenn der MDM-Server (z.B. zu Wartungszwecken) heruntergefahren wird.

Der Client erhält beim Verbindungsabbau (OTS 2-Methodenaufruf onAUntied) im Feld reason im ersten Fall die Begründung „MDM Service Disabled“ und im zweiten Fall die Begründung „MDM Server Shutdown“.

Will der Datengeber die Verbindung abbauen, so sollte er ebenfalls eine entsprechende Begründung im reason-Feld liefern, z.B. „MDM Client Shutdown“ oder „MDM Client Restart“, um detaillierte Logausgaben zu erzeugen.

Zum Verbindungsabbau siehe auch [OTS2] Kapitel 7.6.5 und 7.6.8.

4.5.5 OTS 2 Subscribe

Der Datennehmer nimmt bei Verwendung des OTS 2-Protokolls die Rolle eines OTS 2-Subscriber ein.

Je MDM-Subskription muss eine separate Verbindung zu dem dafür vorgesehenen Service-Endpoint aufgebaut werden (OTS 2-Methode ATie).

Die URL des Service-Endpoints ist folgendermaßen aufgebaut:

```
soap.tls://<BASt-MDM-Broker-Server>/BASt-MDM-OTS2-
DeliveryService/sub/<subscriptionID>
```

Beispiel:

```
soap.tls://broker.mdm-portal.de/BASt-MDM-OTS2-
DeliveryService/sub/2035000
```

Die Verbindungen werden normalerweise dauerhaft – so lange wie möglich – gehalten (und nicht z.B. jede Minute neu aufgebaut). Die Authentisierung findet nur beim Verbindungsaufbau statt (s.u.).

Die nachfolgende Abbildung 13 zeigt einen beispielhaften Ablauf mit Verbindungsaufbau, Bestellung durch den Client, Datenlieferung durch den MDM (hier sind nur zwei Lieferungen eingezeichnet) und Verbindungsabbau durch den Client (ein Verbindungsabbau in umgekehrter Richtung durch den MDM wäre auch möglich). Der Datennehmer (Client) ist auf der linken Seite eingezeichnet.

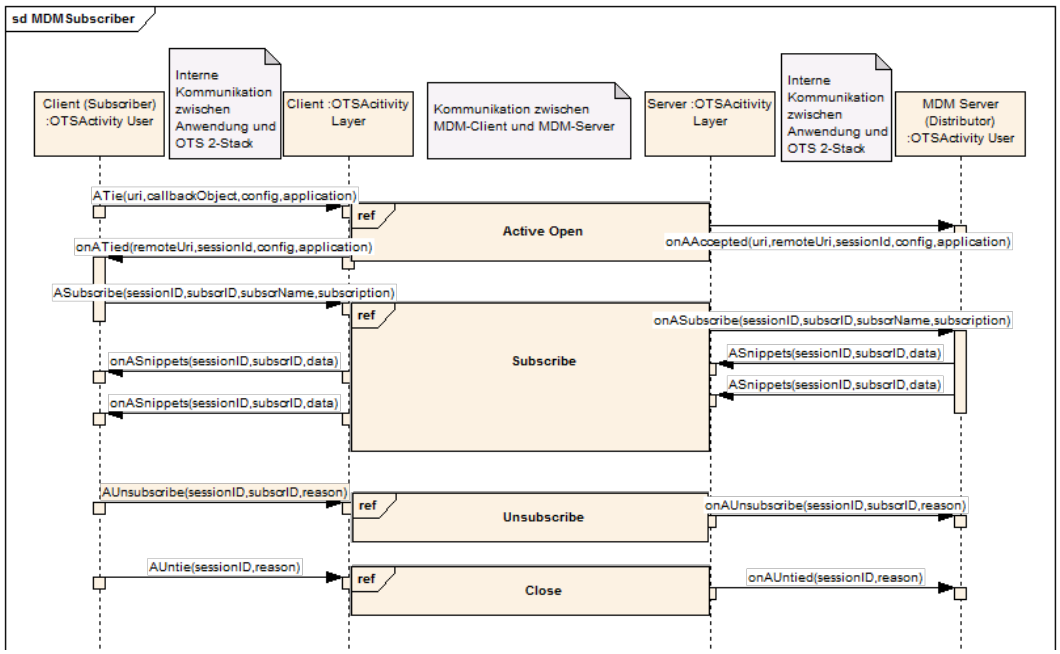

Abbildung 13: Sequenzdiagramm OTS 2-Kommunikation zwischen Datennehmer und MDM

4.5.5.1 Verbindungsaufbau

Das Datennehmersystem muss unter Verwendung seines Maschinenzertifikats eine TLS-Verbindung in Richtung MDM mit der OTS 2-Protokollbindung „SOAP/HTTP mit TLS Verschlüsselung“ aufbauen (OTS 2-Methode ATie, im URL Schema-Feld steht „soap.tls:“).

Der Datennehmer muss in den Konfigurationsinformationen folgende spezielle Features setzen:

- „a_subscriber=1“ (gibt an, dass es sich um einen Datennehmer handelt)
- „a_c_datex_any_mdm=1“ (codiert die speziellen Randbedingungen für den Einsatz des OTS 2-Protokolls im MDM)
- „t_targetURI“ mit der vollständigen Ziel-URI, zum Beispiel „t_targetURI=soap.tls://broker.mdm-portal.de/BASt-MDM-OTS2-DeliveryService/sub/2035000“ (wird vom MDM-Server OTS 2 intern benötigt)

Zum Verbindungsaufbau siehe auch [OTS2] Kapitel 7.6.4.1 und 7.6.4.2.

4.5.5.2 Bestellung

Der Datennehmer muss gemäß OTS 2-Protokoll nach dem erfolgten Verbindungsaufbau (OTS 2-Methodenaufruf onATied) seine Bestellung aufgeben (OTS 2-Methodenaufruf ASubscribe).

Die Bestellung muss vom Typ acSubscriptionAnyType sein. Die gewünschten Daten liegen bereits durch die Auswahl des Service-Endpoints beim MDM fest und werden deshalb in der Bestellung nicht näher spezifiziert.

Es wird empfohlen, im Feld subscrName der OTS 2-Bestellung den MDM-Publikationsnamen und im Feld topic die MDM-Publikations-ID einzutragen. Alle anderen optionalen Felder in der OTS 2-Bestellung entfallen.

Zur Bestellung siehe auch [OTS2] Kapitel 7.6.7.1 und 7.6.7.2.

4.5.5.3 Datenlieferung

Der Datennehmer empfängt nach Aufgabe der Bestellung regelmäßig die Daten des MDM (OTS 2-Methodenaufruf onASnippet). Neue Datenpakete liefert der MDM immer im Sinne eines Push-Verfahrens aus, sobald die Daten vom Datengeber im MDM eintreffen.

Die Datenpakete sind vom Typ acDataAnyType. Die Daten darin werden als BASE64-encodiertes und GZIP-komprimiertes DATEX II-Binärpaket (siehe Kapitel 4.5.3) geliefert.

Zur Datenlieferung siehe auch [OTS2] Kapitel 7.6.7.3 und 7.6.7.4.

4.5.5.4 Verbindungsabbau

Eine bestehende Verbindung kann von beiden Seiten wieder abgebaut werden (OTS 2-Methoden AUnSubscribe zum Beenden der Bestellung bzw. Datenauslieferung und danach AUntie zum Verbindungsabbau).

Vom MDM werden Verbindungen nur abgebaut, wenn eine Publikation oder Subskription aus der Verwaltung (Metadatenverzeichnis) heraus in einen inaktiven Zustand versetzt wird oder wenn der MDM-Server (z.B. zu Wartungszwecken) heruntergefahren wird.

Der Client erhält beim Verbindungsabbau (OTS 2-Methodenaufruf onAUntied) im Feld reason im ersten Fall die Begründung „MDM Service Disabled“ und im zweiten Fall die Begründung „MDM Server Shutdown“.

Will der Datennehmer die Verbindung abbauen, so sollte er ebenfalls eine entsprechende Begründung im reason-Feld liefern, z.B. „MDM Client Shutdown“ oder „MDM Client Restart“, um detaillierte Logausgaben zu erzeugen.

Zum Verbindungsabbau siehe auch [OTS2] Kapitel 7.6.5 und 7.6.8.

4.6 OCIT-C-Schnittstelle

4.6.1 Funktionsumfang

Der MDM implementiert aus dem Funktionsumfang des OCIT-C-Standards die Teilmenge von Protokollfunktionen, die für die Übertragung eines aktuellen Datenpakets mit sämtlichen Informationen einer Publikation erforderlich sind. Gemäß dem Vollständigkeits-Paradigma des MDM wird der Austausch von Teilmengen von Daten (Delta-Lieferungen) nicht unterstützt. Historische Daten können ebenfalls nicht abgefragt werden.

Der MDM implementiert einen Webservice mit der vollständigen WSDL OCIT_Cif.wsdl, der unter dem spezifischen OCIT-Kontext https://broker.mdm-portal.de/BASt-MDM-OCIT-Interface/ocit/ erreichbar ist. Der Aufruf einer nicht unterstützten Operation wird allerdings mit einem SOAP-Fault mit dem Wert „action not supported“ beantwortet.

Das Datenschema wird durch das OCIT-C-Schemata protokoll.xsd definiert. Die OCIT-Nachrichten nutzen zum Transport der Daten eine Datenliste, die mehrere Datenobjekte enthalten kann. In der Kommunikation mit dem MDM darf die Datenliste immer nur genau ein Datenobjekt enthalten. Das DATEX II-Paket muss dabei transparent in das <data>-Element der Nachricht eingebettet werden. Dateneinlieferungen mit mehreren Paketen werden mit einem Fehler quittiert.

Das <data>-Element der OCIT-C-Nachricht ist in der protokoll.xsd als Element vom Typ anyType spezifiziert. Für eine SOAP-konforme Übertragung muss das <data>-Element typisiert werden. Dazu wird ein neuer Datentyp anyD2LogicalModel mit Hilfe der nachstehenden OcitCDatex2.xsd eingeführt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://odg_und_partner/OCIT_C/Datex"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:D2LogicalModel="http://datex2.eu/schema/2/2_0"
targetNamespace="http://odg_und_partner/OCIT_C/Datex"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="d2LogicalModel" type="anyD2LogicalModel"/>
 <xs:complexType name="anyD2LogicalModel">
       <xs:sequence>
             <xs:any namespace="http://datex2.eu/schema/2/2_0"/>
       </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Das Schema kann über die folgende URL referenziert werden: http://bast.s3.amazonaws.com/schema/1446644360562/OcitCDatex2.xsd

4.6.2 Datengeberseite – Publisher Push OCIT-C

Die Funktionalität Publisher Push wird auf die OCIT-C-Methode put abgebildet. Ein put-Aufruf muss immer eindeutig einer Publikation durch Referenzieren einer Publikations-ID zugeordnet werden. Diese Publikations-ID, die vom MDV des MDM automatisch vergeben wird, muss das Datengebersystem im OCIT-C-Element <objectType> übergeben.

Eine put-Nachricht muss genau ein Element vom DATEX II-Typ D2LogicalModel enthalten. Dazu muss der Request eine Datenliste mit genau einem Datenobjekt enthalten. Ein Aufruf mit mehreren Datenobjekten wird vom MDM mit einem Fehler zurückgewiesen. Die Anlieferung eines DATEX II-Elements muss immer vollständig sein, also alle Datenpunkte/ bzw. Objekte der Publikation enthalten. Der MDM wird dies jedoch nicht überprüfen. Es liegt in der Verantwortung des Datengebersystems die Vollständigkeit sicher zu stellen.

Der MDM validiert das DATEX II-Element gegen das im MDM hinterlegte Publikationsschema. Dieses Schema darf nur die DATEX II-Payload ohne den OCIT-C-Container beschreiben. Eine Validierung der ganzen OCIT-Message findet nicht statt. Das Ergebnis der DATEX II-Validierung wird im Log des MDM protokolliert. Das Ergebnis findet keinen Eingang in die OCIT-C-Response.

Der folgende Absatz zeigt beispielhaft eine mögliche Einlieferung im OCIT-C-Format für eine Publikation mit der fiktiven ID=2600103 einer fiktiven Organisation „TEST“. Die DATEX II-Payload ist verkürzt dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <put xmlns="http://odg_und_partner/OCIT_C">
      <userName>Hello</userName>
      <passWord/>
      <objectType>2600103</objectType>
      <putList>
        <putds>
          <identifier>
            <ident>test</ident>
          </identifier>
          <data xsi:type="ns1:anyD2LogicalModel"
xmlns:ns1="http://odg_und_partner/OCIT_C/Datex"
xsi:schemaLocation="http://bast.s3.amazonaws.com/schema/1446644360562/
OcitCDatex2.xsd">
            <ns2:d2LogicalModel modelBaseVersion="2"
extensionName="MDM" extensionVersion="00-01-03"
xmlns:ns2="http://datex2.eu/schema/2/2_0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://bast.s3.amazonaws.com/schema/1370477853100/MDM-
Profile_ParkingFacilityStatus.xsd">
              <ns2:exchange>
                <ns2:supplierIdentification>
                  <ns2:country>de</ns2:country>
                  <ns2:nationalIdentifier>DE-MDM-
TEST</ns2:nationalIdentifier>
                </ns2:supplierIdentification>
              </ns2:exchange>
              <ns2:payloadPublication xsi:type="GenericPublication"
lang="de" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            …
              </ns2:payloadPublication>
            </ns2:d2LogicalModel>
          </data>
        </putds>
      </putList>
    </put>
  </soapenv:Body>
</soapenv:Envelope>
```

Bei der Dateneinlieferung ignoriert der MDM die folgenden Elemente aus dem OCIT-C-Protokoll:

- username
- password
- identifier innerhalb des putds-Attributs

Der MDM quittiert die Einlieferung mit einer OCIT-Meldung vom Typ putResponse. Dabei werden die Elemente folgendermaßen gesetzt:

- lastStart = Zeitpunkt der Einlieferung
- errorCode = 0; Grundsätzlich wird eine formal korrekte Einlieferung immer als fehlerfrei unabhängig von der Qualität des Datenpakets quittiert.
- errorText = ohne Inhalt
- badList = leeres Element

Der folgende Absatz zeigt eine Beispiel-Response.

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <putResponse xmlns="http://odg_und_partner/OCIT_C">
            <lastStart>2015-04-28T11:39:06.948Z</lastStart>
            <errorCode>0</errorCode>
            <errorText></errorText>
            <badList/>
         </putResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

4.6.3 Datennehmerseite – Client Pull OCIT-C

Die Funktionalität Client Pull wird auf die folgenden drei OCIT-C-Methoden abgebildet:

- inquireAll
- get
- wait4Get

Ein OCIT-C-Client kann sich nach seinem Start mit der inquireAll-Methode auf den aktuellen Datenstand synchronisieren. Der MDM unterstützt zu diesem Zweck die inquireAll-Methode. In der inquireAllResponse übergibt der MDM das letzte gültige Paket und dessen MDM-interne ID an den Client. Anschließend kann der Client mit den Methoden get oder wait4Get fortlaufend aktuelle Pakete abholen. Dabei muss der Client jeweils auf seine letzte Paket-ID verweisen. Liegt im MDM kein neues Paket vor, kehrt die get-Methode sofort mit einer leeren Antwort zurück. Die wait4Get-Methode wartet solange, bis ein aktuelles Datenpaket zur Verfügung steht oder ein vom Client vorgegebener oder vom Server definierter maximaler

Timeout erreicht wurde. Durch Nutzung der wait4Get-Methode kann so quasi eine Push-Charakteristik in Richtung Datennehmer implementiert werden. Abweichend zum eigentlichen OCIT-C-Verhalten gibt der MDM mit einer get- bzw. wait4GetResponse immer ein vollständiges Datenpaket zurück und nicht nur Delta-Daten bezogen auf die letzte Position. Der MDM unterstützt prinzipiell keine Deltapakete.

Alternativ zu einem inquireAll-Aufruf kann ein Client auch die get-Methode mit dem Elementwert position=0 aufrufen, um sich zu initialisieren bzw. auf diese Weise jederzeit das letzte verfügbare Paket abzuholen.

Für alle drei Pull-Methoden gilt, dass der MDM die folgenden Elemente des Request aus dem OCIT-C-Protokoll ignoriert:

- username
- password
- watchdog

Das Attribut filterList im Aufruf wird bei allen drei Methoden ebenfalls nicht unterstützt und muss vom Datennehmersystem immer leer angefragt werden.

Ein Client Pull muss immer eindeutig einer Subskription durch Referenzieren einer Subskriptions-ID zugeordnet werden. Diese Subskriptions-ID, die vom MDV des MDM automatisch vergeben wird, muss das Datennehmersystem im OCIT-C-Element <objectType> übergeben.

Der folgende Absatz zeigt beispielhaft eine Anfrage zur Auslieferung im OCIT-C-Format für eine fiktive Subskription mit der ID=2871015 einer fiktiven Organisation „TEST“.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
         <inquireAll xmlns="http://odg_und_partner/OCIT_C">
            <userName>Hello</userName>
            <passWord/>
            <objectType>2871015</objectType>
            <filterList/>
         </inquireAll>
      </soapenv:Body>
   </soapenv:Envelope>
```

Die dazu korrespondierende inquireAllResponse enthält eine Datenliste mit genau einem Element vom DATEX II-Typ D2LogicalModel. Dabei setzt der MDM die nachstehenden OCIT-C-Elemente wie folgt:

- lastStart = ein undefinierter konstanter Zeitpunkt, den der Client ignorieren sollte.
- errorCode = 0
- errorText = ohne Inhalt
- storetime/tstore = Zeitpunkt der Einlieferung der Publikation am MDM
- position = Content-ID des aktuellen Datenpakets
- objectState = modified
- ident = None
- data = DATEX II Payload

Der folgende Absatz zeigt eine Beispiel-Response. Die DATEX II-Payload ist verkürzt dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <inquireAllResponse xmlns="http://odg_und_partner/OCIT_C">
            <lastStart>2015-04-28T11:39:06.948Z</lastStart>
            <errorCode>0</errorCode>
            <errorText></errorText>
            <storetime>2015-04-29T11:57:59.346Z</storetime>
            <position>1</position>
            <dataList>
               <ds>
                  <tstore>2015-04-29T11:57:59.346Z</tstore>
                  <objectState>modified</objectState>
                  <identifier>
                     <ident>None</ident>
                  </identifier>
                  <data xsi:type="ns1:anyD2LogicalModel"
xmlns:ns1="http://odg_und_partner/OCIT_C/Datex" xsi:schemaLocation="
http://odg_und_partner/OCIT_C/Datex
http://bast.s3.amazonaws.com/schema/1446644360562/OcitCDatex2.xsd">
                     <d2LogicalModel modelBaseVersion="2"
extensionName="MDM" extensionVersion="00-01-03"
xmlns="http://datex2.eu/schema/2/2_0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://bast.s3.amazonaws.com/schema/1370439856400/MDM-
Profile_ParkingFacilityStatus.xsd">
                        <exchange>
                           <supplierIdentification>
                              <country>de</country>
                                 <nationalIdentifier>DE-MDM-
TEST</nationalIdentifier>
                           </supplierIdentification>
                        </exchange>
                        <payloadPublication
xsi:type="GenericPublication" lang="de"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                              …
                        </payloadPublication>
                     </d2LogicalModel>
                  </data>
               </ds>
            </dataList>
         </inquireAllResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Mit Hilfe des Elements <position> aus der inquireAllResponse kann das Datennehmersystem im Folgenden die get- oder wait4Get-Methode parametrieren, um Folgepakete zu lesen.

Ein get-Aufruf muss immer eindeutig einer Subskription durch Referenzieren einer Subskriptions-ID und einem Datenpaket durch Referenzieren der Content-ID zugeordnet werden. Diese Subskriptions-ID muss das Datennehmersystem im OCIT-C-Attribut <objectType> übergeben, die Content-ID im Attribut <position>. Ein get-Aufruf unter Nutzung von Start- und Endezeit unterstützt der MDM nicht.

Der MDM bildet die getResponse und die wait4GetResponse unter Nutzung der gleichen Attribute wie in der inquireAllResponse.

Für den wait4Get-Aufruf gelten die gleichen Anforderungen wie für den regulären get-Aufruf. Ergänzend muss das Datennehmersystem im Element <maxWaitTime> den Timeout-Wert des Client übermitteln. Liegt dieser Wert über dem im MDM konfigurierten Maximalwert, so wird der MDM-Timeout angewendet. Die Möglichkeit verschiedene Objekte mit einem einzigen wait4Get-Aufruf zu lesen, wird vom MDM nicht unterstüzt. Mit einem wait4Get-Aufruf kann also immer nur eine einzige Subskription abgefragt werden. Listen-Abfragen werden mit einem Fehler zurückgewiesen.

Die Abgabe von Datenpaketen am MDM erfolgt grundsätzlich komprimiert. Dabei kommt die GZIP-Komprimierung zum Einsatz. Dies gilt auch bei der Auslieferung mit dem OCIT-C-Protokoll. Datennehmersysteme müssen die Pakete daher im Webserver dekomprimieren, bevor diese mit Hilfe des OCIT-C-Protokolls weiter verarbeitet werden können.

4.6.4 Fehlerbehandlung

Folgende OCIT-C-Fehlercodes werden verwendet:

Fehlercode	Beschreibung
access error (1)	grundsätzlich fehlerhafte Parametrierung des Requests
internal error (22)	Fehler bei der internen Verarbeitung des Requests
missing parameters to execute the method (23)	fehlende subscription id bei get,wait4get oder inquireAll

Tabelle 10: verwendete OCIT-C Fehlercodes

5 Zertifikatsbasierte M2M-Kommunikation

Die Security-Komponente der MDM-Plattform erfordert einen zertifikatsbasierten Datenaustausch zwischen Datengebersystem und Plattform einerseits sowie zwischen Plattform und Datennehmersystem.

Dieses Kapitel gibt zunächst einen Überblick über die Funktionen der Security-Komponente und beschreibt anschließend die Schritte, die Datengeber und Datennehmer ausführen müssen, um Zertifikate anzufordern und für die M2M-Kommunikation einzurichten.

Das Zertifikat wird nach der Beantragung erstellt und dem Datengeber/-nehmer per E-Mail zugesandt. Das zur Signatur erforderliche Passwort wird per SMS übermittelt.

Datengebersystem/-nehmersystem müssen abschließend das Zertifikat in ihre IT-Infrastruktur einbinden, so dass der Datenaustausch mit der MDM-Plattform authentisiert erfolgen kann.

5.1 Aufgaben der Security-Komponente

Die Security-Komponente ist für die Realisierung der sicherheitstechnischen Aspekte der MDM-Plattform verantwortlich. Dazu gehört insbesondere die Authentisierung von Datengebersystemen und Datennehmersystemen, die mit der MDM-Plattform kommunizieren wollen.

Bevor die an der MDM-Plattform ankommenden Datenpakete angenommen werden, muss deren Herkunft überprüft werden. Dazu gehört die Authentisierung des zum Datenpaket gehörigen Datengebersystems mittels digitalen Zertifikats. Jedes Datengebersystem muss über ein gültiges Zertifikat verfügen, mit dem es sich an der Plattform anmeldet. Die Security-Komponente authentifiziert das vom Datengebersystem gesendete Zertifikat innerhalb der MDM-Plattform.

Bevor ein Datenpaket an ein Datennehmersystem gesendet wird, muss die Identität des Datennehmersystems überprüft werden. Jedes Datennehmersystem muss sich mittels digitalen Zertifikats an der MDM-Plattform authentisieren. Die Security-Komponente authentifiziert das vom Datennehmersystem gesendete Zertifikat innerhalb der MDM-Plattform.

Die Vertraulichkeit der Kommunikation zwischen Datengebersystem und MDM-Plattform einerseits und MDM-Plattform und Datennehmersystem andererseits, muss durch die ausschließliche Verwendung einer SSL/TLS-Transportverschlüsselung gewährleistet werden.

Die Security-Komponente setzt standardkonforme [X.509v3]-Zertifikate für die Authentisierung voraus; siehe auch [PKI]. Die

Zertifikate müssen technisch über einen clientseitigen, zertifikatsbasierten Verbindungsaufbau in die HTTPS-Verbindung zu den Datennehmer- und Datengebersystemen eingebunden werden. Die präsentierten Zertifikate werden auf Gültigkeit und gegen eine Sperrliste geprüft.

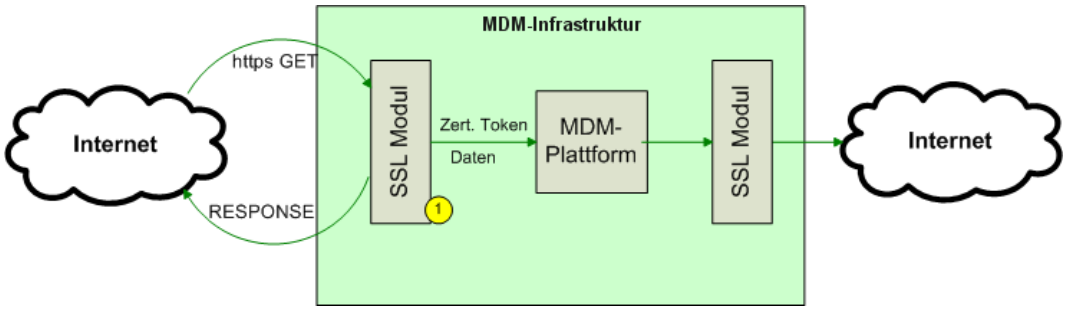

Abbildung 14: Übersicht Sicherheitsarchitektur

Das SSL-Modul 1 in Abbildung 14 schickt für vorgegebene URLs eine Zertifikatsanfrage an den Sender und prüft das daraufhin erhaltene Zertifikat auf Gültigkeit und gegen eine Sperrliste. Anschließend leitet es das Zertifikat an die Security-Komponente der MDM-Plattform weiter.

5.2 Hinweis zu Server Name Indication

Die MDM-Plattform unterstützt keine Server Name Indication (SNI).

Dies hat zur Folge, dass Datengeber für das Client-Pull-Verfahren und Datennehmer für das Publisher-Push-Verfahren keine virtuellen Server für die M2M-Kommunikation verwenden können. Jede registrierte Maschine kann nur eine eindeutige IP-Adresse repräsentieren.

5.3 Maschinenzertifikat beantragen

Der Betreiber der MDM-Plattform vermittelt zwischen Datengeber- bzw. Datennehmersystem und dem Zertifikatsaussteller. Datengeber und Datennehmer beantragen dafür im Rahmen ihrer Registrierung ein oder mehrere Maschinenzertifikate über die Administrations-GUI der MDM-Plattform. Das Zertifikat wird ihnen anschließend allerdings von der zertifikatausstellenden Organisation zugesendet, nicht vom Betreiber der MDM-Plattform.

Um ein Maschinenzertifikat anfordern zu können, müssen Sie mit Ihrer Organisation bereits auf der MDM-Plattform registriert sein.

Wie Sie ein Maschinenzertifikat über die MDM-Plattform beantragen ist im [BHB] beschrieben.

5.4 Maschinenzertifikat und Ausstellerzertifikat installieren

Im Apache-Webserver binden Sie das Maschinenzertifikat folgendermaßen ein:

```
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
```

Den zugehörigen private key tragen Sie wie folgt ein:

```
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.crt/server.key
```

Zusätzlich müssen Sie das Ausstellerzertifikat im Webserver hinterlegen:

```
SSLCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-
client.crt
```

Das Zertifikat ist über den Key mit dem Passwort verschlüsselt, das Ihnen per SMS mitgeteilt wurde. Verwenden Sie das Passwort zum Entschlüsseln.

Weitere Erläuterungen zu diesen Direktiven finden Sie in der mod_ssl-Dokumentation:

http://httpd.apache.org/docs/current/mod/mod_ssl.html#sslcertificatefile

http://httpd.apache.org/docs/current/mod/mod_ssl.html#sslcacertificatefile

Hinweis: Wenn Sie das Maschinenzertifikat und das Ausstellerzertifikat innerhalb einer gemeinsamen p12-Datei erhalten, müssen Sie beide Zertifikate aus dieser Datei extrahieren und anschließend installieren. Die Anleitung hierzu finden Sie in Kapitel 8.1

5.5 Authentifizierung der MDM-Plattform als Webclient

Fungiert die MDM-Plattform in der M2M-Kommunikation als Webclient, so authentifiziert sie sich mit ihrem Serverzertifikat, sofern der Webserver auf Datengeber- oder Datennehmerseite diese Option aktiviert hat. Datengeber- und Datennehmersysteme sollten diese Option aktivieren und das Zertifikat verifizieren, um festzustellen, dass die Requests tatsächlich von der MDM-Plattform abgesetzt wurden.

Die für die Verifikation erforderlichen CA-Zertifikate können unter https://service.mdm-portal.de/doc/MDM-CA-Bundle.zip

heruntergeladen werden und müssen im Datengeber- bzw. Datennehmersystem hinterlegt werden.

Hinweis: Verwenden Sie nicht das MDM-Serverzertifikat für die Verifikation. Dieses wird regelmäßig ausgetauscht.

5.6 Authentifizierung von Datengeber/Datennehmer-Webclients

Fungiert das Datengeber- oder Datennehmersystem in der M2M-Kommunikation als Webclient, so muss dieser sich mit seinem Maschinenzertifikat gegenüber der MDM-Plattform authentifizieren. Die Plattform akzeptiert nur Requests von Systemen, die im Metadatenverzeichnis registriert sind. Aufgrund des Zertifikats kann die Maschine der Organisation zugewiesen werden. Des Weiteren kann geprüft werden, ob die Organisation Eigentümer der Publikation bzw. der Subskription ist, für die ein Datenaustausch stattfinden soll.

Das Serverzertifikat für broker.mdm-portal.de wurde von Comodo erstellt. In den meisten Fällen wird es nicht nötig sein, das Comodo CA Zertifikat zu installieren, wenn der „Truststore“ des Betriebssystems benutzt wird. Trotzdem kann es notwendig sein, die CA Zertifikate der MDM CA zu installieren. Ein Archiv mit allen benötigten Zertifikaten finden sie unter:

https://service.mdm-portal.de/doc/MDM-CA-Bundle.zip

6 Ausnahmen und Fehlermeldungen

6.1 Ausnahme – Unveränderte Daten

Falls ein DATEX II Client-Pull-Request das Header Field “If-Modified-Since” nutzt, und keine aktuelleren als die bereits abgerufenen Datenpakete vorliegen, wird ein HTTP Statuscode 304 = "Not-Modified" erzeugt. Gleiches gilt, falls noch gar keine Daten vorliegen.

6.2 Fehlermeldungen bei SOAP-Requests

Fehlermeldungen bei SOAP-Requests werden als SOAP-Fault gemeldet.

Hierbei wird die Fehlermeldung im "faultstring" der folgenden SOAP-Response gesendet:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>...</faultstring>
      </S:Fault>
   </S:Body>
</S:Envelope>
```

6.3 Fehlermeldungen bei HTTPS-Requests

Falls kein SOAP-Request vorliegt, wird die Fehlermeldung mit HTTP 503 und Content-Type "text" gesendet.

6.4 Fehlerbehandlung im Rahmen des OTS 2-Protokolls

Nachfolgend werden verschiedene Fehlersituationen beschrieben, die für einen OTS 2-Client auftreten können. Weitere Informationen zu den OTS 2-Standardfehlermeldungen (z.B. enthaltene Parameter) finden sich in [OTS2].

6.4.1 Sitzungsaufbau

Falls die erforderlichen Zertifikate fehlen oder ungültig sind, wird auf Client-Seite ein Fehler vom Typ 1301 (TConnect failed) ausgelöst. Dies geschieht entweder durch den Methodenaufruf onError oder onRemoteError, je nachdem, ob der Fehler bereits auf Client- oder erst auf Server-Seite festgestellt wird. Im Feld reason des Fehlers wird als Grund „Certificate error“ angegeben.

Falls die Zertifikate unpassend sind (gültig, aber z.B. nicht passend zur gewünschten Publikation/Subskription), wird ebenfalls ein Fehler 1301

(TConnect failed) ausgelöst. Im Feld reason steht dann „Certificate inappropriate“.

Falls versucht wird, eine falsche Protokollbindung zu verwenden (nicht soap.tls), wird ein Fehler vom Typ 1001 (invalid URI) per onError ausgelöst.

Falls in den Konfigurationsinformationen das Feature „t_targetURI“ fehlt, wird ein Fehler vom Typ 5102 (rejected) per onRemoteError ausgelöst. Im Feld reason steht „Feature t_targetURI required“.

Falls in den Konfigurationsinformationen das Feature „a_c_datex_any_mdm“ fehlt, wird ein Fehler vom Typ 3301 (ATie failed) per onRemoteError ausgelöst. Im Feld reason steht „Feature a_c_datex_any_mdm required“.

Falls der Server keine Verbindungen annimmt oder nicht erreichbar ist, wird ein Fehler vom Typ 1301 (TConnect failed) ausgelöst. Im Feld reason steht „Unavailable URI”.

6.4.2 Bestellung

Falls die Bestellung vom falschen Typ ist (nicht acSubscriptionAnyType), wird ein Fehler vom Typ 8709 (Subscription: invalid parameter) ausgelöst. Im Feld par steht der ungültige Typ.

6.4.3 Datenlieferung

Falls eine Datenlieferung an den MDM vom falschen Typ ist (nicht acDataAnyType oder nicht verarbeitbarer Inhalt), wird ein MDM-spezifischer Fehler vom Typ 10001 ausgelöst.

6.4.4 Allgemein

Falls über längere Zeit keine Kommunikation stattfindet und die Verbindung in einen Timeout läuft oder falls die Verbindung aus anderen Gründen unerwartet unterbrochen wird, wird ein Fehler vom Typ 1003 (Transport connection lost) ausgelöst.

Scheitert eine Datenübertragung, wird ein Fehler 1501 (TSendData failed) ausgelöst.

In beiden Fällen muss die Verbindung beendet und ggf. ein Neuaufbau versucht werden.

Ein interner Fehler der MDM-Brokerkomponente wird mit dem MDM-spezifischen Fehler vom Typ 10002 angezeigt.

7 Beispiele

7.1 HTTPS-Schnittstelle

7.1.1 Datengeber Client Pull HTTPS (Container)

Die Publikations-ID muss in der URL als Parameter übergeben werden.

Request:

```
GET https://<DG-Server>/<Context>?publicationID=2053008
content-type: text/plain
accept-encoding: identity,gzip
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<container xmlns="http://ws.bast.de/container/TrafficDataService"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:ns3="http://www.w3.org/2000/09/xmldsig#">
    <header>
        <Identifier>
            <publicationId>2053008</publicationId>
        </Identifier>
    </header>
    <body>
        <binary id="test-id-bin" type="hexBinary">
            &lt;![CDATA[]]&gt;
        </binary>
        <xml schema="test-schema" id="test-id-xml">
              <n4:musterDatenRoot>
                    <n4:trafficData origin="home" />
              </n4:musterDatenRoot>
        </xml>
    </body>
</container>
```

7.1.2 Datengeber Publisher Push HTTPS (Container)

Die Publikations-ID ist in den XML-Daten enthalten.

```
<?xml version='1.0' encoding='UTF-8'?>
<ns3:containerRootElementEl xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:ns3="http://ws.bast.de/container/TrafficDataService">
  <ns3:header>
    <ns3:Identifier>
      <ns3:publicationId>12345</ns3:publicationId>
    </ns3:Identifier>
  </ns3:header>
  <ns3:body>
    <ns3:binary id="test-id-bin" type="hexBinary">
      dGVzdC10ZXh0&#xD;.
```

```
    </ns3:binary>
    <ns3:xml schema="test-schema" id="test-id-xml">
      <n4:musterDatenRoot>
      <n4:trafficData origin="home"/>
      </n4:musterDatenRoot>
    </ns3:xml>
  </ns3:body>
</ns3:containerRootElementEl>
```

7.1.3 Datennehmer Client Pull HTTPS (DATEX II)

Der Request muss keine weiteren Daten enthalten. Die Subskriptions-ID muss im Pfad der URL und zusätzlich als Parameter übergeben werden.

```
GET https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000000/clientPullService?subscriptionID=2000000
```

7.1.4 Datennehmer Client Pull HTTPS (Container)

Der Request muss keine weiteren Daten enthalten. Die Subskriptions-ID muss in der URL als Parameter übergeben werden.

Request:

```
GET https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/container/v1.0?subscriptionID=2000000
```

7.2 SOAP-Schnittstelle

7.2.1 Datengeber Publisher Push SOAP (DATEX II)

Die Publikations-ID muss im Pfad der URL übergeben werden.

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000002/supplierPushService
```

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <d2LogicalModel xmlns="http://datex2.eu/schema/2/2_0"
modelBaseVersion="2">
      <exchange>

<subscriptionReference>subscriptionReference</subscriptionReference>
        <supplierIdentification>
          <country>de</country>
          <nationalIdentifier>TestClient</nationalIdentifier>
          <internationalIdentifierExtension/>
        </supplierIdentification>
      </exchange>
      <payloadPublication xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="SituationPublication" lang="DE">
        <feedDescription>
```

```
          <values>
            <value lang="DE">test-test</value>
          </values>
        </feedDescription>
        <feedType>feedType</feedType>
        <publicationTime>2011-03-
02T10:36:34.336+01:00</publicationTime>
        <publicationCreator>
          <country>de</country>
          <nationalIdentifier>TestClient</nationalIdentifier>
          <internationalIdentifierExtension/>
        </publicationCreator>
        <situation version="0.1" id="GUID-Mattst-1299058594339">
          <overallSeverity>none</overallSeverity>
          <headerInformation>
            <areaOfInterest>regional</areaOfInterest>
          </headerInformation>
          <situationRecord xsi:type="AnimalPresenceObstruction">
            <generalPublicComment>
              <comment>
                <values>
                  <value lang="DE"></value>
                </values>
              </comment>
            </generalPublicComment>
          </situationRecord>
        </situation>
      </payloadPublication>
    </d2LogicalModel>
  </S:Body>
</S:Envelope>
```

7.2.2 Datengeber Client Push SOAP (Container)

Die Publikations-ID ist in den XML-Daten enthalten.

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns3:container xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:ns3="http://ws.bast.de/container/TrafficDataService">
      <ns3:header>
        <ns3:Identifier>
          <ns3:publicationId>12345</ns3:publicationId>
        </ns3:Identifier>
      </ns3:header>
      <ns3:body>
        <ns3:binary id="test-id-bin"
type="hexBinary">dGVzdC10ZXh0&#xD;.</ns3:binary>
        <ns3:xmlschema="test-schema" id="test-id-xml"/>
      </ns3:body>
    </ns3:container>
  </S:Body>
```

```
</S:Envelope>
```

7.2.3 Datennehmer Client Pull SOAP (DATEX II)

Der Request muss keine weiteren Daten enthalten. Die Subskriptions-ID muss im Pfad der URL übergeben werden.

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/srv/2000000/clientPullService
```

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body />
</S:Envelope>
```

7.2.4 Datennehmer Client Pull SOAP (Container)

Die Subskriptions-ID ist in den XML-Daten enthalten.

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns3:pullContainerDataClientRequestEl
xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ns2="http://schemas.xmlsoap.org/ws/2002/07/utility"
xmlns:ns3="http://ws.bast.de/container/TrafficDataService">
      <ns3:subscriptionId>2000000</ns3:subscriptionId>
    </ns3:pullContainerDataClientRequestEl>
  </S:Body>
</S:Envelope>
```

7.2.5 Datennehmer Publisher Push SOAP (DATEX II)

Erwartete Response vom Datennehmersystem:

```
<D2LogicalModel:d2LogicalModel modelBaseVersion="2"
xsi:schemaLocation="http://datex2.eu/schema/2/2_0/
DATEXIISchema_2_2_0.xsd"
xmlns:D2LogicalModel="http://datex2.eu/schema/2/2_0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<D2LogicalModel:exchange>
        <D2LogicalModel:response>acknowledge</D2LogicalModel:response>
 </D2LogicalModel:exchange>
...
</D2LogicalModel:d2LogicalModel>
```

7.3 OTS 2-Schnittstelle

7.3.1 Protokollbeispiel SOAP

Der Ablauf bei Verwendung des OTS 2-Protokolls wird beispielhaft am Kommunikationsprotokoll eines Datennehmers dargestellt.

Für einen Datengeber ist der Ablauf i.W. gleich mit der Ausnahme, dass Bestellung (aSubscribe) und Datenlieferung (aSnippets) in der umgekehrten Richtung übermittelt werden (die Bestellung wird über ein tGetR empfangen und die Datenlieferung über ein tSend verschickt anstatt umgekehrt wie im dargestellten Beispiel).

7.3.1.1 Verbindungsaufbau

Die Subskriptions-ID muss im Pfad der URL und die URL zusätzlich als Konfigurationsparameter t_targetURI übergeben werden.

Request (tConnect):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
       <tConnect xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <clientId>ff97c2b4c0a8013800d902546789ad4c</clientId>
              <username/>
              <password/>
              <localTransportId>1</localTransportId>
              <timeout>100000</timeout>
              <neededConfig version="m_configListClient_C3X">
                     <cfgs>
                            <cfg>
                                   <name>t_targetURI=
soap.tls://broker.mdm-portal.de/BASt-MDM-OTS2-
DeliveryService/sub/2035000</name>
                                   <min>0</min>
                                   <max>0</max>
                            </cfg>
                     </cfgs>
              </neededConfig>
       </tConnect>
 </S:Body>
</S:Envelope>
```

Response (tConnectR):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body>
       <tConnectR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <transportId>
```

```
                    <clientPart>1</clientPart>
                    <serverPart>27</serverPart>
                </transportId>
                <config version="m_configListServer_S3A"/>
        </tConnectR>
  </env:Body>
</env:Envelope>
```

Request (tGet, tGetR Response s.u.):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
        <tGet xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
                <transportId>
                    <clientPart>1</clientPart>
                    <serverPart>27</serverPart>
                </transportId>
        </tGet>
  </S:Body>
</S:Envelope>
```

Request (sOpen über tSend, tSendR Response ist leer):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
        <tSend xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
                <transportId>
                    <clientPart>1</clientPart>
                    <serverPart>27</serverPart>
                </transportId>
                <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="sOpenType">
                    <neededConfig version="m_configListClient_C2X"/>
                </data>
        </tSend>
  </S:Body>
</S:Envelope>
```

Response (sOpenResponse über tGetR, tGet Request s.o.):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
        <tGetR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
                <ds>
                    <tSend>
```

```
                              <transportId>
                                    <clientPart>1</clientPart>
                                    <serverPart>27</serverPart>
                              </transportId>
                              <data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="sOpenResponseType">
                                    <sessionId>266419a0-1d02-11e1-a7c2-
000c294483b2</sessionId>
                                    <config
version="m_configListServer_S2A"/>
                              </data>
                        </tSend>
                </ds>
        </tGetR>
 </env:Body>
</env:Envelope>
```

Request (tGet, tGetR Response s.u.):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
        <tGet xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
                <transportId>
                      <clientPart>1</clientPart>
                      <serverPart>27</serverPart>
                </transportId>
        </tGet>
 </S:Body>
</S:Envelope>
```

Request (aTie über sMsg über tSend, tSendR Response ist leer):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
        <tSend xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
                <transportId>
                      <clientPart>1</clientPart>
                      <serverPart>27</serverPart>
                </transportId>
                <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="sMsgType">
                      <msg xsi:type="aTieType">
                            <application xsi:type="acApplicationType">

 <appVersion>OTS2TestClient_V_1.0.0</appVersion>
                            </application>
```

```
                              <neededConfig
version="configListCounterPart">
                                    <cfgs>
                                          <cfg>
                                                <name>s_layer</name>
                                                <min>1</min>
                                                <max>1</max>
                                          </cfg>
                                          <cfg>

 <name>a_distributor_sub</name>
                                                <min>1</min>
                                                <max>1</max>
                                          </cfg>
                                          <cfg>

 <name>a_subscriber</name>
                                                <min>0</min>
                                                <max>0</max>
                                          </cfg>
                                          <cfg>

 <name>a_c_datex_any_mdm</name>
                                                <min>0</min>
                                                <max>0</max>
                                          </cfg>
                                          <cfg>
                                                <name>t_layer</name>
                                                <min>1</min>
                                                <max>1</max>
                                          </cfg>
                                          <cfg>
                                                <name>a_layer</name>
                                                <min>1</min>
                                                <max>1</max>
                                          </cfg>
                                    </cfgs>
                              </neededConfig>
                        </msg>
                  </data>
            </tSend>
 </S:Body>
</S:Envelope>
```

Response (aTieResponse über sMsg über tGetR, tGet Request s.o.):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body>
       <tGetR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
             <ds>
```

```
                    <tSend>
                          <transportId>
                                <clientPart>1</clientPart>
                                <serverPart>27</serverPart>
                          </transportId>
                          <data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="sMsgType">
                                <msg xsi:type="aTieResponseType">
                                      <application
xsi:type="acApplicationType">

 <appVersion>GUI_OTS2_ActivityLayer_SRVTest_V_1.0.0</appVersion>
                                      </application>
                                      <config
version="configListCounterPart">
                                            <cfgs>
                                                  <cfg>

 <name>s_layer</name>

 <min>1</min>

 <max>1</max>
                                                  </cfg>
                                                  <cfg>

 <name>a_distributor_sub</name>

 <min>1</min>

 <max>1</max>
                                                  </cfg>
                                                  <cfg>

 <name>a_subscriber</name>

 <min>0</min>

 <max>0</max>
                                                  </cfg>
                                                  <cfg>

 <name>a_c_datex_any_mdm</name>

 <min>0</min>

 <max>0</max>
                                                  </cfg>
                                                  <cfg>

 <name>t_layer</name>

 <min>1</min>
```

```
 <max>1</max>
                                                          </cfg>
                                                          <cfg>

 <name>a_layer</name>

 <min>1</min>

 <max>1</max>
                                                          </cfg>
                                                    </cfgs>
                                              </config>
                                        </msg>
                                  </data>
                            </tSend>
                      </ds>
               </tGetR>
 </env:Body>
</env:Envelope>
```

7.3.1.2 Datenbestellung

Im Feld topic wird empfohlen, die MDM-Publikations-ID einzutragen.

Request (aSubscribe über sMsg über tSend, tSendR Response ist leer):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
       <tSend xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <transportId>
                     <clientPart>1</clientPart>
                     <serverPart>27</serverPart>
              </transportId>
              <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="sMsgType">
                     <msg xsi:type="aSubscribeType">
                            <subscrId>1322843092</subscrId>
                            <subscrName>OTS2TestClient</subscrName>
                            <subscription
xsi:type="acSubscriptionAnyType">
                                   <topic>2035000</topic>
                            </subscription>
                     </msg>
              </data>
       </tSend>
 </S:Body>
</S:Envelope>
```

7.3.1.3 Datenlieferung

Request (tGet, tGetR Response s.u.):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
       <tGet xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <transportId>
                     <clientPart>1</clientPart>
                     <serverPart>27</serverPart>
              </transportId>
       </tGet>
 </S:Body>
</S:Envelope>
```

Response (aSnippets über sMsg über tGetR, tGet Request s.o.):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body>
       <tGetR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <ds>
                     <tSend>
                            <transportId>
                                   <clientPart>1</clientPart>
                                   <serverPart>27</serverPart>
                            </transportId>
                            <data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="sMsgType">
                                   <msg xsi:type="aSnippetsType">

 <subscrId>1322843092</subscrId>
                                          <data
xsi:type="acDataAnyType">
                                                 <dataAny
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:type="xs:string">&lt;binary
type=&quot;base64BinaryDatex2Gzip&quot;
id=&quot;0&quot;&gt;H4sIAAAAAAAAADTdCZYkOa5D0S3Z5Dbsf2PJ+xTZ/U//qswIdx
skigQB8H6f4/f7Xfdz7dd3/I59v87ffpzPfZy/69uf+zvO...</dataAny>
                                          </data>
                                   </msg>
                            </data>
                     </tSend>
              </ds>
       </tGetR>
 </env:Body>
</env:Envelope>
```

7.3.1.4 Abbestellung

Request (aUnsubscribe über sMsg über tSend, tSendR Response ist leer):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
       <tSend xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <transportId>
                     <clientPart>1</clientPart>
                     <serverPart>27</serverPart>
              </transportId>
              <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="sMsgType">
                     <msg xsi:type="aUnsubscribeType">
                           <subscrId>1322843092</subscrId>
                           <reason>OTS2TestClient closes</reason>
                     </msg>
              </data>
       </tSend>
 </S:Body>
</S:Envelope>
```

7.3.1.5 Verbindungsabbau

Request (sClose über tSend, tSendR Response ist leer):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
       <tSend xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
              <transportId>
                     <clientPart>1</clientPart>
                     <serverPart>27</serverPart>
              </transportId>
              <data xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="sCloseType">
                     <sessionId>266419a0-1d02-11e1-a7c2-
000c294483b2</sessionId>
                     <reason>End TestClient</reason>
              </data>
       </tSend>
 </S:Body>
</S:Envelope>
```

Response (sCloseResponse über tSend über tGetR, tGet Request nicht gezeigt):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body>
```

```
        <tGetR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
               <ds>
                      <tSend>
                             <transportId>
                                    <clientPart>1</clientPart>
                                    <serverPart>27</serverPart>
                             </transportId>
                             <data
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="sCloseResponseType">
                                    <sessionId>266419a0-1d02-11e1-a7c2-
000c294483b2</sessionId>
                             </data>
                      </tSend>
               </ds>
        </tGetR>
 </env:Body>
</env:Envelope>
```

Request (tDisconnect):

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
        <tDisconnect xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
               <transportId>
                      <clientPart>1</clientPart>
                      <serverPart>27</serverPart>
               </transportId>
               <reason>close</reason>
        </tDisconnect>
 </S:Body>
</S:Envelope>
```

Response (tDisconnectR):

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header/>
 <env:Body>
        <tDisconnectR xmlns="http://opentrafficsystems.org/OTS2"
xmlns:ns2="http://datex2.eu/schema/2_0RC2/2_0" xmlns:ns3="http://otec-
konsortium.de/OCIT-I_OITD">
               <rTransportId>
                      <clientPart>1</clientPart>
                      <serverPart>27</serverPart>
               </rTransportId>
        </tDisconnectR>
 </env:Body>
</env:Envelope>
```

8 Anhang A

8.1 p12-Datei für Apache Server Konfiguration aufbereiten

Die Apache-Serverkonfiguration kann keine Dateien vom Typ p12 verarbeiten. Für die Aufbereitung sind manuelle Schritte erforderlich, die in folgendem Kapitel beschrieben werden:

Exportieren Sie zunächst die Schlüssel und Zertifikate. Führen Sie in der Kommandozeile folgenden Befehl aus:

```
openssl.exe pkcs12 -in <p12-Datei> -out <sammeldatei.pem>
```

Beispiel:

```
openssl.exe pkcs12 -in ehp.otten-software.de.p12 -out ehp.otten-
software.de.keyandcerts.pem
```

Geben Sie in der Openssl-Umgebung die Zertifikats-Passwörter ein:

```
>Enter Import Password:     <Passwort aus der SMS>
>MAC verified OK
>Enter PEM pass phrase:     <selbst vergebene Passphrase für den
Schlüssel>
>Verifying - Enter PEM pass phrase: <Wiederholung der selbst
vergebenen Passphrase für den Schlüssel>
```

Öffnen Sie die Datei <sammeldatei.pem> mit einem Texteditor:

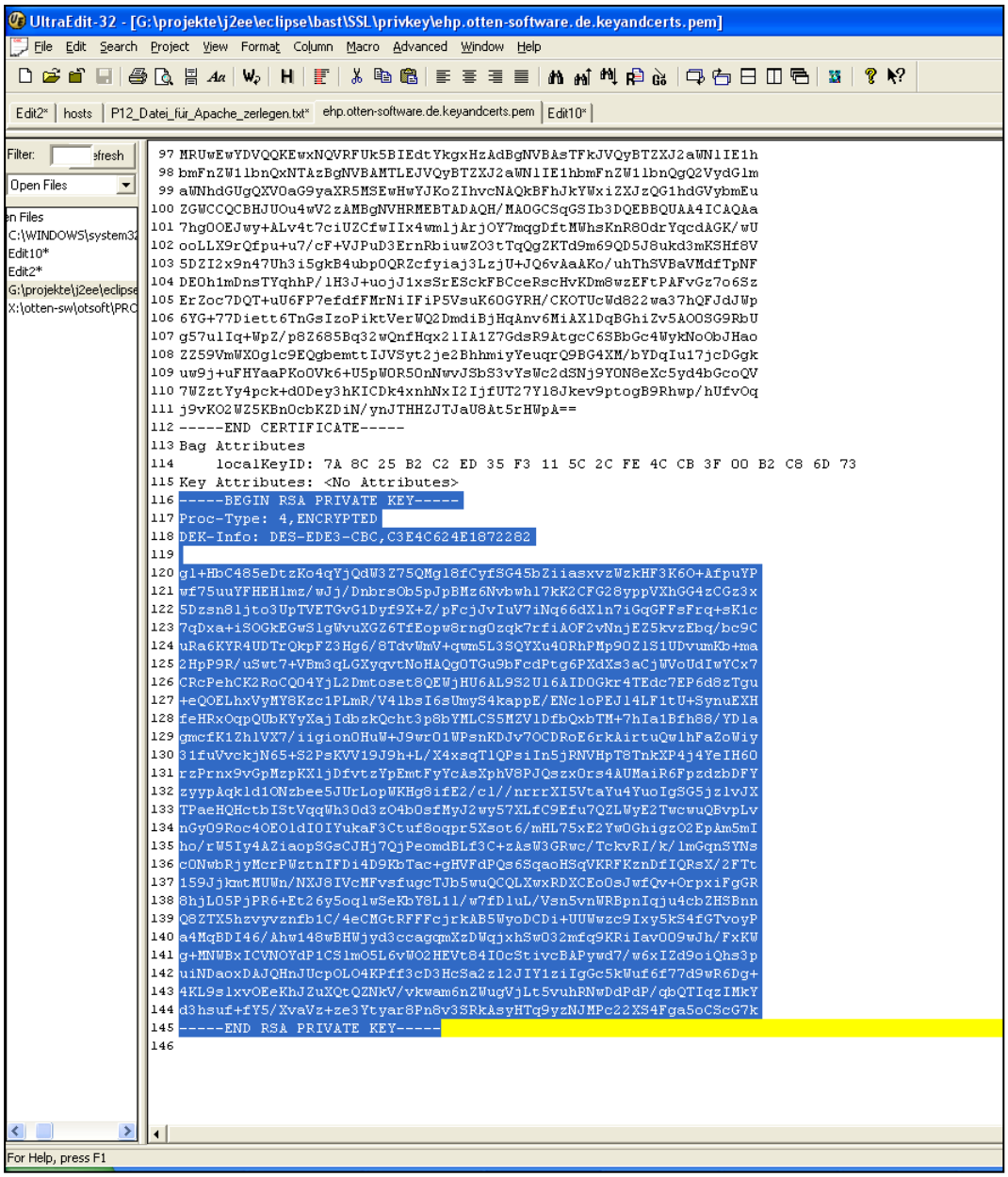

Abbildung 15: Datei <sammeldatei.pem>

Kopieren Sie den Teil von

```
--- BEGIN RSA PRIVATE KEY ----
```

bis

```
---END RSA PRIVATE KEY ---
```

in eine neue Datei namens <server.key>

Entfernen Sie die Passphrase, um zu verhindern, dass diese bei jedem Neustart des Servers angefordert wird:

```
openssl rsa -in <server.key> -out <server.key.nopass >
```

Beispiel:

```
openssl rsa -in server.key -out ehp.otten-software.de.key
> Enter pass phrase for server.key:      <Die zuvor selbst vergebene
Passphrase eintragen>
>writing RSA key
```

Tragen Sie die erzeugte .key-Datei in der Apache-Konfiguration unter folgendem Attribut ein:

```
SSLCertificateKeyFile
```

Als nächsten Schritt teilen Sie die Zertifikate in zwei Dateien auf. Öffnen Sie dazu zunächst die Datei <sammeldatei.pem> mit einem Texteditor:

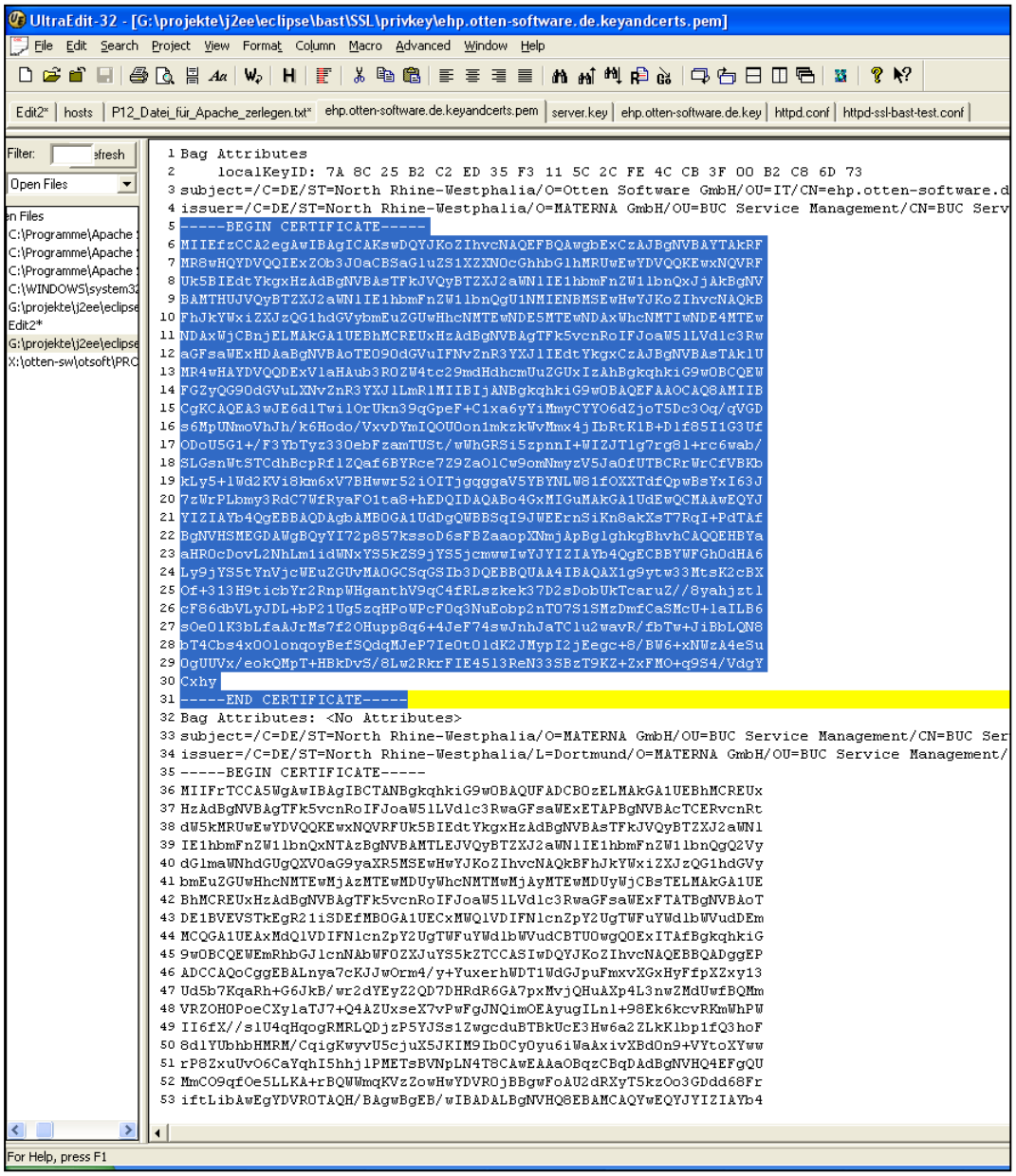

Abbildung 16: Datei <sammeldatei.pem>

Kopieren Sie das Serverzertifikat in eine neue Textdatei <server.crt>.

Tragen Sie diese Datei in der Apache Konfiguration unter folgendem Attribut ein:

```
SSLCertificateFile
```

Kopieren Sie die verbleibenden Zertifikate in eine neue Textdatei <ca-cert-chain.crt>.

Tragen Sie diese Datei in der Apache-Konfiguration unter folgendem Attribut ein:

```
SSLCertificateChainFile
```

Tragen Sie das MDM-Client-Zertifikat inkl. Zertifikatshierarchie unter folgendem Apache-Attribut ein:

```
SSLCACertificateFile
```

Beispiel einer Apache-Konfiguration:

```
SSLCertificateFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.crt\ehp.otten-software.de.crt"
SSLCertificateKeyFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.key\ehp.otten-software.de.key"
SSLCertificateChainFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.crt\bast_cert_chain.crt"
SSLCACertificateFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.crt\bast_trust_chain.crt"
```