Technische Schnittstellen-beschreibung

Version 3.0.1 – 24.08.2021

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 potenzielle 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.

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 OCIT-C-Protokoll angeboten.

1.2 Gliederung des Dokuments

Das Dokument ist in die folgenden Abschnitte gegliedert:

- Abschnitt 1 enthält eine kurze Übersicht, 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 grundlegend beschrieben.
- Abschnitt 5 beschreibt im Detail das Format DATEX II v2.
- Abschnitt 6 beschreibt entsprechend DATEX II v3.
- Abschnitt 7 beschreibt das MDM-eigene Container-Format.
- Abschnitt 8 beschreibt die Maßnahmen, mit der die M2M-Kommunikation abgesichert wird.

1.3 Referenzierte Dokumente

1.3.1 Allgemein

[Quelle]	Herausgeber
[BHB]	MDM-Benutzerhandbuch, V2.9.0 https://service.mdm-portal.de/doc/MDM-Benutzerhandbuch.pdf
[GZIP]	RFC 1952 (Mai 1996) GZIP File Format Specification Version 4.3, https://tools.ietf.org/rfc/rfc1952.txt
[HTTP/1.1]	RFC 2616 (Juni 1999) Hypertext Transfer Protocol -- HTTP/1.1 https://www.ietf.org/rfc/rfc2616.txt
[HTTPS]	RFC 2818 (Mai 2000) HTTP over TLS https://www.ietf.org/rfc/rfc2818.txt
[MCS]	MDM Containerformat Spezifikation https://www.mdm-portal.de/wp-content/uploads/2019/03/mdm_spezifikation_container-v1.1.pdf
[OCIT-C]	OCIT-C Spezifikation Version 1.1_R1 vom 30.10.2014 https://www.ocit.org/media/ocit-c_protokoll_v1.1_r1.pdf https://www.mdm-portal.de/files/ocit.wsdl
[PKI]	RFC 2459 (Januar 1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile https://www.ietf.org/rfc/rfc2459.txt
[SOAP]	SOAP Version 1.2 https://www.w3.org/TR/soap12-part1/

[Quelle]	Herausgeber
[URL]	RFC 1738 (Dezember 1994) Uniform Resource Locators (URL) https://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. https://www.itu.int/rec/T-REC-X.509-199708-S/en

Tabelle 1: Referenzierte Dokumente (übergreifend)

1.3.2 DATEX II v2

[Quelle]	Herausgeber
[DATEXIIv2PSM]	DATEX II v2.0 Exchange Platform Specific Model
[DATEXIIv2Pull]	DATEX II v2.0 Pull wsdl https://www.mdm-portal.de/files/Pull.wsdl
[DATEXIIv2Push]	DATEX II v2.0 Push wsdl https://www.mdm-portal.de/files/Push.wsdl
[DATEXIIv2Schema]	DATEX II XML Schema 2.0
[DATEXIIv2SDG]	DATEX II v2.0 Software Developers Guide, Version v.1.2
[DATEXIIv2Spec]	Umfasst die folgenden Dokumente, die auf https://www.datex2.eu allen registrierten Benutzern zum Download bereitstehen: [DATEXIIv2PSM], [DATEXIIv2UG]
[DATEXIIv2UG]	DATEX II v2.0 User Guide v.1.2

Tabelle 2: Referenzierte Dokumente (DATEX II v2)

1.3.3 DATEX II v3

[Quelle]	Herausgeber
[DATEXIIv3Annex]	Annexes to Platform Specific Model: https://docs.datex2.eu/exchange/2020/psm/annexes.html
[DATEXIIv3Pull]	DATEX II v3.0 Snapshot Pull wsdl https://www.mdm-portal.de/files/SnapshotPull.wsdl
[DATEXIIv3Push]	DATEX II v3.0 Snapshot Push wsdl https://www.mdm-portal.de/files/SnapshotPush.wsdl
[DATEXIIv3Exc]	Download für Level A- und Level B-Publikationen: https://www.mdm-portal.de/files/Exchange2020LevelAB.zip Download für Level C-Publikationen: https://www.mdm-portal.de/files/Exchange2020LevelC.zip

[Quelle]	Herausgeber
[DATEXIIv3ExcUG]	Exchange 2020 User Guide : https://docs.datex2.eu/exchange/2020/userguide/

Tabelle 3: Referenzierte Dokumente (DATEX II v3)

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.
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
JSSE	Java Secure Socket Extension
M2M	Machine-to-Machine
MDM	Mobilitätsdatenmarktplatz
MDV	Metadatenverzeichnis
OCIT	Open Communication Interface for Road Traffic Control 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

Abkürzung	Auflösung
WS	Webserver
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

Tabelle 4: Abkürzungsverzeichnis

2 Komponenten der MDM-Plattform im Überblick

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

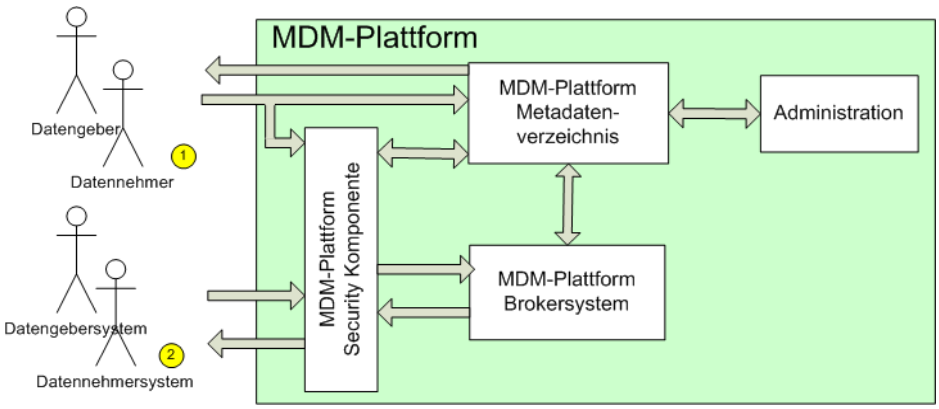

Abbildung 1: Komponenten der MDM-Plattform

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 5: 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 nicht überprüft. Die Web-GUI des MDM erlaubt über eine Schaltfläche das jeweils aktuell im MDM gespeicherte Datenpaket einer Publikation zu validieren. Dazu werden die Dateischemata über die URL bezogen, die in der Publikationsbeschreibung hinterlegt ist. Für Publikationen im DATEX II Format liegt es in der Verantwortung des Datengebers, die korrekten Dateischemata 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 von deren Validität und liefert diese auch an die Datennehmer aus, wenn es nicht gegenüber den hinterlegten Schemata valide sein 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 [DATEXIIv2Spec]. Die ursprüngliche Implementierung der MDM-Plattform beruht auf DATEX II v2. Aktuell werden sowohl v2 als auch v3 unterstützt.

DATEX II definiert XML-Strukturen für den Austausch von Mobilitätsdaten. Das zugrunde liegende Schema kann unter https://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 DATEX II zugrunde liegenden Dokumente sind im Kapitel 1.3 Referenzierte Dokumente als [DATEXIIv2Spec] 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 den Datengebern die Skalierbarkeitsbürde 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 in einem Header-Element zu übertragen, die insbesondere zur Steuerung des Kommunikationsprozesses benutzt werden.

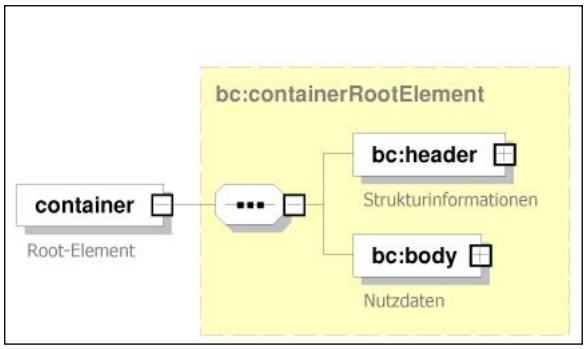

Abbildung 2: Übersicht Containerformat

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 Datengebersystem 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.

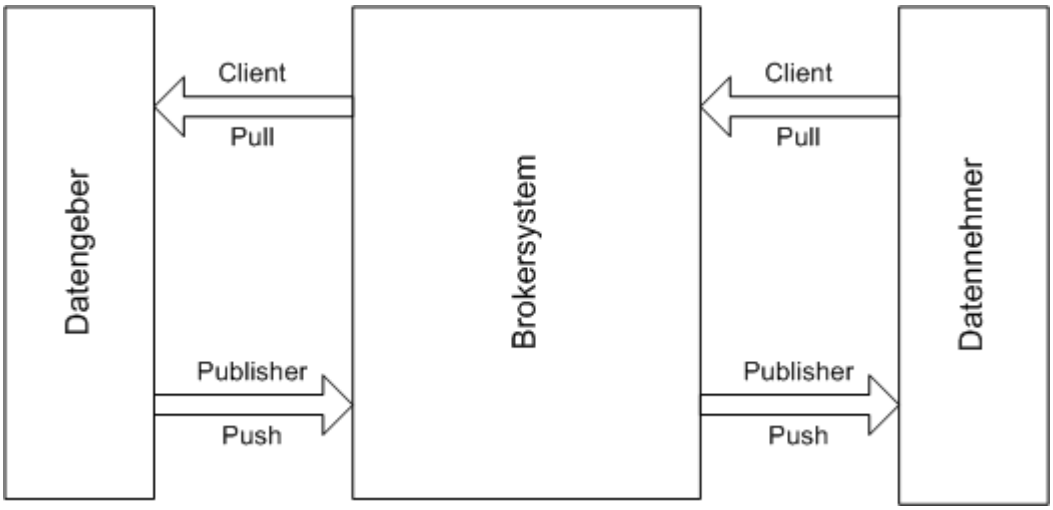

Abbildung 3: Schnittstellen zwischen Datengeber, Brokersystem und Datennehmer

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 wird zudem das OCIT-C-Protokoll unterstützt.

Tabelle 6 zeigt, welche Kommunikationswege unterstützt werden. Dabei ist für jedes Datenformat (DATEX II / Container), Kommunikationsmuster (Client Pull / Publisher Push) und Protokoll (HTTPS, SOAP, 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		
		SOAP	HTTPS	OCIT	SOAP	HTTPS	OCIT
DATEX II v2	Client Pull	5.1.1.1 Server	5.2.1.1 Server	-	5.1.2.1 Client	5.2.2.1 Client	5.3.3 Client
	Publisher Push	5.1.1.2 Client	-	5.3.2 Client	5.1.2.2 Server	-	-
DATEX II v3	Client Pull	6.2.1.1 Server	6.3.1.1 Server	-	6.2.2.1 Client	6.3.2.1 Client	-
	Publisher Push	6.2.1.2 Client	-	-	6.2.2.2 Server	-	-
Container	Client Pull	7.1.1.1 Server	7.2.1.1 Server	-	7.1.2.1 Client	7.2.2.1 Client	-
	Publisher Push	7.1.1.2 Client	7.2.1.2 Client	-	7.1.2.2 Server	7.2.2.2 Server	-

Tabelle 6: 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.

Grundsätzlich stehen Datenpakete für Datennehmer nur während der in der Publikationsdefinition hinterlegten Gültigkeit zur Abholung zur Verfügung. Nach Ablauf dieser Frist wird das Paket aus dem Puffer gelöscht.

Solange kein neues Datenpaket angeliefert wird, erhält das Datennehmersystem eine Fehlermeldung vom Typ „No Content“, der je nach Protokoll unterschiedlich ausgeprägt ist. Näheres dazu ist bei den Protokollen beschrieben.

4.1 Verschlüsselung der Kommunikation

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 die Clientauthentisierung kommen X.509-konforme Zertifikate zum Einsatz [PKI], die vom Betreiber der MDM-Plattform ausgestellt werden.

4.2 Komprimierung

Bei der Datenübermittlung zwischen MDM-Plattform und Datengebersystemen können sowohl GZIP-encodierte (d. h. komprimierte) als auch unkomprimierte HTTPS-Requests und -Responses verwendet werden.

Die Datenübermittlung zwischen MDM-Plattform und Datennehmersystemen findet – abweichend zu [DATEXIIv2PSM] – immer mittels GZIP-encodierter HTTPS-Requests und -Responses statt.

Dies gilt unabhängig vom gewählten Exchange-Protokoll für HTTP, SOAP und OCIT-C.

4.3 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 (DATEX II v2) bzw. am sog. messageContainer (DATEX II v3) 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).

Es folgt je ein Beispiel für DATEX II v2 und DATEX II v3 einer Implementierung unter Einhaltung des Unveränderlichkeitsverspechens.

4.3.1 DATEX II v2

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="https://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <d2LogicalModel xmlns="https://datex2.eu/schema/2/2_0" modelBaseVersion="2">
      <exchange>
        <supplierIdentification>
          <country>de</country>
          <nationalIdentifier>DE-MDM-Musterorg</nationalIdentifier>
        </supplierIdentification>
      </exchange>
      <payloadPublication xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
                          xsi:type="SituationPublication" lang="DE">
        <publicationTime>2021-08-18T13:09:00.106+02:00</publicationTime>
        ...
      </payloadPublication>
    </d2LogicalModel>
  </S:Body>
</S:Envelope>
```

4.3.2 DATEX II v3

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Body>
  <con:messageContainer xmlns:con="http://datex2.eu/schema/3/messageContainer"
      xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
      xmlns:d2="http://datex2.eu/schema/3/d2Payload"
      xmlns:loc="http://datex2.eu/schema/3/locationReferencing"
      xmlns:com="http://datex2.eu/schema/3/common"
      xmlns:sit="http://datex2.eu/schema/3/situation"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://datex2.eu/schema/3/messageContainer
./DATEXII_3_MessageContainer.xsd"
           modelBaseVersion="3">
    <con:payload lang="en"
           xsi:type="sit:SituationPublication"
           modelBaseVersion="3">
    ...
    </con:payload>
    <con:exchangeInformation modelBaseVersion="3">
      <ex:exchangeContext>
        <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
        <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
        <ex:supplierOrCisRequester/>
      </ex:exchangeContext>
      <ex:dynamicInformation>
        <ex:exchangeStatus>online</ex:exchangeStatus>
        <ex:messageGenerationTimestamp>2021-07-21T13:00:00
        </ex:messageGenerationTimestamp>
      </ex:dynamicInformation>
    </con:exchangeInformation>
  </con:messageContainer>
</S:Body>
```

i	**Wichtige Hinweise zu DATEX II v3**

1. Das Element codedExchangeProtocol (erforderliches Element), ist abhängig vom verwendeten Ein/Auslieferungs-Protokoll.
2. Bei jeder Auslieferung des MDM an das Datennehmersystem wird der Wert dieses Elements wie folgt gesetzt:
   a. „snapshotPush“, wenn der MDM das Datenpaket an den Datennehmer mittels SOAP-Push ausliefert
   b. „snapshotPull“, wenn der Datennehmer eine SOAP-Pull Anfrage an den MDM schickt
   c. „snapshotPull“, wenn der Datennehmer eine HTTP-Pull Anfrage an den MDM schickt
3. Der Wert des Elements messageGenerationTimestamp wird bei der Auslieferung des MDM an das Datennehmersystem nicht neu gesetzt. Das heißt, der Timestamp ist ein Ende-zu-Ende-Wert, der dem Original-Timestamp des eingelieferten Datenpakets entspricht.

4.4 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 7: Operationsmodi des MDM

Für einen Datenaustausch mit dem MDM muss grundsätzlich für alle Protokolle eine Transportverschlüsselung mit TLS 1.2 und eine Authentifizierung mittels standardkonformer X.509v3-Zertifikate verwendet werden. Sofern die Standardprotokolle eine Basic-Authentication mittels Benutzername und Passwort vorsehen, werden diese Protokollelemente ignoriert. Dies gilt insbesondere für das OCIT-Protokoll [OCIT-C], wie nachfolgend noch erläutert wird.

Der MDM implementiert eine OCIT-C-Schnittstelle auf Basis des OCIT-C-Standards in der Version 1.1_R1 vom 30.10.2014. Der OCIT-C-Funktionsumfang wird durch den MDM dabei nur eingeschränkt und unter der Vorgabe einer spezifischen Nutzung von Protokollelementen angeboten. Als Datenmodell wird bei OCIT-C nur DATEX II v2 verwendet wie im vorliegenden Dokument bzw. in [DATEXIIv2Spec] beschrieben. Die OCIT-C Datenmodelle werden nicht unterstützt.

Hinweis: Das vormals durch den MDM unterstützte Protokoll „OTS 2“ ist seit 01. Januar 2021 nicht mehr verfügbar.

4.4.1 Datengeberseite

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

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

4.4.2 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 OCIT-C-Protokolls agiert das Brokersystem als Server und das Datennehmersystem als Client.

4.5 Nutzung des „If-Modified-Since“ Header-Feldes im HTTPS-Protokoll

Das Brokersystem unterstützt das Header-Field “If-Modified-Since" in Verbindung mit dem Feld „Last Modified“ (vgl. [HTTP/1.1]). Hierdurch wird der wiederholte Versand bzw. die wiederholte Abholung bereits zugestellter Nachrichten 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
```

4.5.1 Datengeber

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.

Das Datengebersystem sollte dieses Header-Field immer setzen, um die MDM-Plattform in die Lage zu versetzen, dieses Feature anwenden zu können!

4.5.2 Datennehmer

Die Responses des Brokersystems enthalten immer das Header-Field „Last-Modified“. Falls das Datennehmersystem dieses Feature nutzen möchte, muss es immer den Wert aus dem letzten Last-Modified Header-Field mitsenden.

Es wird dringend empfohlen, dieses Feature auf Datennehmerseite zu implementieren!

4.5.3 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.

5 DATEX II v2

5.1 SOAP-Schnittstelle

5.1.1 Datengeberseite

5.1.1.1 Client Pull SOAP

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

5.1.1.1.1 Anbieten eines Webservices

Das Datengebersystem muss einen Webservice mit der Methode getDatex2Data anbieten, der aufgrund der DATEX II Pull WSDL [DATEXIIv2Pull] definiert ist. Als Input wird dabei nichts erwartet, als Output bekommt das MDM-Brokersystem die angeforderten Daten im DATEX II Format zurück: im body-Element wird ein Objekt vom Typ d2LogicalModel erwartet.

Über die MDM-Administrations-Komponente muss der Datengeber die URL seines Service-Endpoints in der Publikations-Konfiguration hinterlegen.

5.1.1.1.2 Aufrufen eines Webservices

Das MDM-Brokersystem stellt einen aufgrund der DATEX II Pull WSDL [DATEXIIv2Pull] definierten Webservice-Client zum Aufruf von Webservices bereit. Dieser Webservice muss Daten gemäß einem DATEX II v2 Schema [DATEXIIv2Schema] zurückliefern. Es wird erwartet, dass von dem Gesamtschema ein geeignetes Profil zum Einsatz kommt.

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 potenzielle Datennehmer in entsprechenden Paketpuffern zwischengespeichert. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

5.1.1.2 Publisher Push SOAP

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-Brokersystems unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

5.1.1.2.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice mit der Methode putDatex2Data an, der aufgrund der Spezifikation DATEX II Push WSDL [DATEXIIv2Push] definiert ist. Als Input werden die zu überliefernden Daten erwartet, als Output bekommt das Datengebersystem Bestätigungsdaten im DATEX II Format zurück. Dabei wird jeweils im body-Element ein Objekt vom Typ d2LogicalModel erwartet.

Der Output besteht aus einer Bestätigung des Empfangs.

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

Die URL ist folgendermaßen aufgebaut:

```
https://broker.mdm-portal.de/BASt-MDM-Interface/srv/<publication ID>/supplierPushService
```

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="https://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <d2LogicalModel xmlns="https://datex2.eu/schema/2/2_0" modelBaseVersion="2">
      <exchange>
        <supplierIdentification>
          <country>de</country>
          <nationalIdentifier>DE-MDM-Musterorg</nationalIdentifier>
        </supplierIdentification>
      </exchange>
      <payloadPublication xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
                          xsi:type="SituationPublication" lang="DE">
        <publicationTime>2021-08-18T13:09:00.106+02:00</publicationTime>
        ...
      </payloadPublication>
    </d2LogicalModel>
  </S:Body>
</S:Envelope>
```

5.1.1.2.2 Aufrufen des Webservices

Das Datengebersystem muss einen aufgrund der DATEX II Snapshot Push WSDL [DATEXIIv2Push] definierten Webservice Client zum Aufruf des Webservices bereitstellen. Der Webservice muss am Publikations-spezifischen Service-Endpoint des MDM-Brokersystems die Daten anliefern. Die zu verwendende URL des Service-Endpoints wird in der Publikations-Konfiguration des MDM-Portals angezeigt. Das MDM-Brokersystem nimmt diese Daten an und speichert sie in einem Paketpuffer. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

5.1.2 Datennehmerseite

5.1.2.1 Client Pull SOAP

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

5.1.2.1.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice mit der Methode getDatex2Data an, der aufgrund der Spezifikation [DATEXIIv2Pull] 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. Im body-Element kann ein Objekt vom Typ d2LogicalModel erwartet. Aufgrund der

übermittelten Subskriptions-ID kann die MDM-Plattform den zugehörigen Paketpuffer sowie das Datenpaket ermitteln.

Beispiel:

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

5.1.2.1.2 Aufrufen des Webservices

Das Datennehmersystem muss einen aufgrund der Spezifikation [DATEXIIv2Pull] 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://broker.mdm-portal.de/BASt-MDM-Interface/srv/<Subskriptions-ID>/clientPullService
```

5.1.2.2 Publisher Push SOAP

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.

5.1.2.2.1 Anbieten eines Webservices

Das Datennehmersystem muss einen Webservice mit der Methode putDatex2Data anbieten, der aufgrund der Spezifikation [DATEXIIv2Push] definiert ist. Als Input werden die angeforderten Daten erwartet, als Output bekommt die MDM-Plattform Bestätigungsdaten im DATEX II Format zurück. Dabei wird jeweils im body-Element ein Objekt vom Typ d2LogicalModel erwartet. Das Format des Input-Parameters entspricht dem DATEX II Schema [DATEXIIv2Schema].

5.1.2.2.2 Aufrufen des Webservices

Das MDM-Brokersystem stellt einen auf Basis von [DATEXIIv2Push] 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. Das nachstehende Beispiel zeigt den Inhalt eines solchen sog. Acknowledge-Response, welche bei Übertragung mit dem SOAP-Protokoll im Body-Element enthalten ist.

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

5.2 HTTPS-Schnittstelle

5.2.1 Datengeberseite

5.2.1.1 Client Pull HTTPS

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 [DATEXIIv2PSM], Kapitel 4, die Punkte C.1–C.12.

Dabei ist zu berücksichtigen, dass die weiteren, optionalen Regeln keine Anwendung finden. Die Optionen zur Authentisierung (C.13, C.14, C.17) finden keine Anwendung, da sie bei der Verwendung des für den MDM verpflichtenden HTTPS-Verfahrens obsolet sind. C.18-C.27 entfallen, da die Optionen sich nur auf die optionale Bereitstellung von DATEX II Daten in Dateiform beziehen, die beim MDM nicht zur Anwendung kommt. Siehe hierzu auch Anhang B – DATEX II HTTP Protokollunterstützung.

5.2.1.1.1 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.

Beachten Sie auch die Hinweise in Kapitel 4.5, Nutzung des „If-Modified-Since“ Header-Feldes.

5.2.1.1.2 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 [DATEXIIv2PSM] Abschnitt 4 hat die Response den Content–Type „text/xml; charset=utf-8“ und kann 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.

5.2.2 Datennehmerseite

5.2.2.1 Client Pull HTTPS

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

5.2.2.1.1 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 Subskriptions-ID muss im Pfad der URL und zusätzlich als Parameter übergeben werden. Die URL des Brokersystems ist daher wie folgt aufgebaut:

```
https://broker.mdm-portal.de/BASt-MDM-Interface/srv/<Subskriptions-ID>/clientPullService?subscriptionID=<Subskriptions-ID>
```

Beachten Sie auch die Hinweise in Kapitel 4.5, Nutzung des „If-Modified-Since“ Header-Feldes.

5.2.2.1.2 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 [DATEXIIv2PSM] Abschnitt 4 hat die Response den Content Type „text/xml; charset=utf-8“ und wird – abweichend zu [DATEXIIv2PSM] – immer GZIP-komprimiert verschickt.

Als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 8 beschriebenen Bedeutungen gelten.

	Beschreibung
Request	Request GET /BASt-MDM-Interface/srv/clientPullService?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx < d2LogicalModel > … </d2LogicalModel >
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 304: Not Modified, die angeforderte Ressource wird nicht erneut übertragen - 403: Authentifizierungsfehler - 404: Kein Datenpaket im Paketpuffer zur Subskription, Keine Subskription, oder eine nicht mehr gültige Subskription zum Subskriptionsparameter gefunden - 503: Service Unavailable (z. B. bei Wartung)

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

5.3 OCIT-C-Schnittstelle

Hinweis: Diese Schnittstelle unterstützt nur DATEX II v2. Sie unterstützt nicht den DATEX II v3 Standard!

5.3.1 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. Datenanlieferungen 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="https://odg_und_partner/OCIT_C/Datex"
           xmlns:xs="https://www.w3.org/2001/XMLSchema"
           xmlns:D2LogicalModel="https://datex2.eu/schema/2/2_0"
           targetNamespace="https://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="https://datex2.eu/schema/2/2_0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

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

5.3.2 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.

In der Metadatenverwaltung des MDM kann das DATEX II Element manuell gegen das im MDM hinterlegte Publikationsschema validiert werden. Dieses Schema darf nur die DATEX II Payload ohne den OCIT-C-Container beschreiben. Eine Validierung der ganzen OCIT-Message findet nicht statt.

Der folgende Absatz zeigt beispielhaft eine mögliche Anlieferung 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="https://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <put xmlns="https://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="https://odg_und_partner/OCIT_C/Datex"
                xsi:schemaLocation="https://bast.s3.amazonaws.com/schema/
                                    1446644360562/OcitCDatex2.xsd">
            <ns2:d2LogicalModel modelBaseVersion="2" extensionName="MDM"
                                extensionVersion="00-01-03"
                                xmlns:ns2="https://datex2.eu/schema/2/2_0"
                                xmlns:xsi="https://www.w3.org/2001/
                                           XMLSchema-instance"
                                xsi:schemaLocation="
                                 https://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="https://www.w3.org/2001/
                                               XMLSchema-instance">
            …
            </ns2:payloadPublication>
          </ns2:d2LogicalModel>
        </data>
      </putds>
    </putList>
  </put>
 </soapenv:Body>
</soapenv:Envelope>
```

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

- username
- password
- identifier innerhalb des putds-Attributs

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

- lastStart = Zeitpunkt der Anlieferung
- errorCode = 0; Grundsätzlich wird eine formal korrekte Anlieferung 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="https://schemas.xmlsoap.org/soap/envelope/"
                    xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                    xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
     <soapenv:Body>
        <putResponse xmlns="https://odg_und_partner/OCIT_C">
           <lastStart>2015-04-28T11:39:06.948Z</lastStart>
           <errorCode>0</errorCode>
           <errorText></errorText>
           <badList/>
        </putResponse>
     </soapenv:Body>
  </soapenv:Envelope>
```

5.3.3 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 Requests 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="https://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <inquireAll xmlns="https://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 des 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 Anlieferung 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="https://schemas.xmlsoap.org/soap/envelope/"
                     xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                     xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <inquireAllResponse xmlns="https://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="https://odg_und_partner/OCIT_C/Datex"
```

```
                    xsi:schemaLocation=" https://odg_und_partner/OCIT_C/Datex
                                         https://bast.s3.amazonaws.com/
                                         schema/1446644360562/OcitCDatex2.xsd">
                <d2LogicalModel modelBaseVersion="2" extensionName="MDM"
                                extensionVersion="00-01-03"
                                xmlns="https://datex2.eu/schema/2/2_0"
                                xmlns:xsi="https://www.w3.org/2001/
                                           XMLSchema-instance"
                                xsi:schemaLocation="
                          https://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="https://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.

Das folgende Beispiel zeigt eine get-Anfrage zur Auslieferung im OCIT-C-Format für eine fiktive Subskription mit der ID=2871015 und der fiktiven Vorgänger-Content-ID=3876098:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
        <get xmlns="https://odg_und_partner/OCIT_C">
           <objectType>2871015</objectType>
           <position>3876098</position>
        </get>
     </soapenv:Body>
  </soapenv:Envelope>
```

Der MDM bildet daraufhin die getResponse und analog eine 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 sollte das Datennehmersystem im Element <maxWaitTime> den Timeout-Wert des Clients übermitteln. Wird dieses Element nicht übermittelt, verhält sich der wait4Get-Aufruf wie ein regulärer get-Aufruf und kehrt sofort mit einer leeren Antwort zurück, sollte zwischenzeitlich kein neues Datenpaket angeliefert worden sein. Liegt dieser Wert über dem im MDM konfigurierten Maximalwert von 120 Sekunden, so wird der MDM-Default-Timeout angewendet, und der Aufrufer erhält spätestens nach 120 Sekunden eine Antwort. Diese ist leer, wenn innerhalb der Wartezeit kein neues Datenpaket angeliefert wurde.

Die Möglichkeit, verschiedene Objekte mit einem einzigen wait4Get-Aufruf zu lesen, wird vom MDM nicht unterstützt. Mit einem wait4Get-Aufruf kann also immer nur eine einzige Subskription abgefragt werden. Listen-Abfragen werden mit einem Fehler zurückgewiesen.

Der folgende Absatz zeigt beispielhaft eine wait4Get-Anfrage zur Auslieferung im OCIT-C-Format für eine fiktive Subskription mit der ID=2871015, der fiktiven Content-ID=3876098 und dem Wert maxWaitTime=60 Sekunden.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
      <wait4Get xmlns=\"http://odg_und_partner/OCIT_C\"  maxWaitTime='60'>
       <get xmlns=\"http://odg_und_partner/OCIT_C\">
           <objectType>2871015</objectType>
           <position>3876098</position>
       </get>
       </wait4Get>
  </soapenv:Body>
</soapenv:Envelope>
```

Nachstehend eine beispielhafte wait4Get-Antwort des MDM:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
<soapenv:Header/>
  <soapenv:Body>
    <ocit:wait4GetResponse xmlns:ocit="http://odg_und_partner/OCIT_C">
    <ocit:lastStart>2021-07-22T15:37:38.000+02:00</ocit:lastStart>
    <ocit:errorCode>0</ocit:errorCode>
    <ocit:errorText></ocit:errorText>
    <ocit:waitResponseList>
      <ocit:storetime>2021-08-02T11:53:51.480+02:00</ocit:storetime>
      <ocit:objectType>0</ocit:objectType>
      <ocit:position>0</ocit:position>
      <ocit:dataList>
        <ocit:ds>
          <ocit:tstore>2021-08-02T11:53:51.480+02:00</ocit:tstore>
          <ocit:objectState>modified</ocit:objectState>
          <ocit:identifier>
            <ocit:ident>None</ocit:ident>
          </ocit:identifier>
```

```
          <ocit: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">
          </ocit:data>
        </ocit:ds>
      </ocit:dataList>
    </ocit:waitResponseList>
    </ocit:wait4GetResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

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 weiterverarbeitet werden können.

5.3.4 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 9: verwendete OCIT-C Fehlercodes

6 DATEX II v3

Gegenüber DATEX II v2 Exchange geschieht der Datentransport in DATEX II v3 Exchange 2020 über eine MessageContainer-Struktur. Da nicht alle laut DATEX II v3 Spezifikation möglichen Elemente dieser Struktur im MDM verwendet werden, wird hier von einem Minimal-MessageContainer gesprochen. Dieser muss neben einem payload-Element auch ein exchangeInformation-Element enthalten. Der Minimal-MessageContainer sowie das exchangeInformation-Element sind unter [DATEXIIv3Exc] verfügbar.

Das exchangeInformation-Element besteht aus zwei Datenstrukturen mit den folgenden Pflichtattributen:

- exchangeContext:
  - codedExchangeprotocol: Ein Attribut von einem Enumerationstyp. Der Wert unterscheidet sich abhängig vom angewendeten Protokoll:
    - Für SOAP Schnittstellen wird hier entweder „snapshotPull“ oder „snapshotPush“ verwendet
    - Für HTTP pull wird hier „snapshotPull“ verwendet.
  - exchangeSpecificationVersion: Der MDM erwartet hier den Wert „3.0“.
  - supplierOrCisRequester: Um konform mit dem Standard zu sein, muss hier ein leeres XML-Element aufgenommen werden.
- dynamicInformation:
  - exchangeStatus: Hier wird der Wert „online“ fest erwartet.
  - messageGenerationTimestamp: aktuelle Uhrzeit der Meldungsgenerierung.

6.1 Hinweise zur Behandlung von XML-Schemas mit Exchange 2020

Will man eine Publikation für DATEX II v3 Inhalte einrichten, muss der Datengeber mehrere XML-Schemas auf zwei Ebenen bereitstellen:

1. **Inhaltsdaten:** DATEX II v3 hat das Konzept von Namensräumen (engl. Namespace) in DATEX II eingeführt. Wenn man das Datenprofil für eine Publikation erzeugt, wird für jeden Namensraum ein gesondertes XML-Schema erzeugt. Jede Instanz der Publikation muss das Einstiegsschema referenzieren, welches je nach Kompatibilitätsstufe entweder DATEXII_3_D2Payload.xsd (Level A oder B) oder LevelC_3_D2Payload.xsd (Level C) heißt. Dieses Schema importiert alle weiteren XML-Schemas des jeweiligen Datenprofils.
2. **Protokolldaten:** Zum Transport von DATEX II v3 Inhalten mit der korrespondierenden DATEX II Exchange 2020-Spezifikation, wird das Schema der Inhaltsdaten bei den auf dem MDM umgesetzten Optionen in zwei weitere XML-Schemas eingebettet: MessageContainer.xsd und ExchangeInformation.xsd.

Die beiden Protokolldatenschemas wurden für die Anwendung im Rahmen des MDM profiliert. Wichtig ist, dass das Schema des DATEX II MessageContainer Objekts die Stelle ist, an der die Protokolldaten mit den Inhaltsdaten verknüpft werden. Dieses Schema muss also angepasst werden, je nachdem ob Level A oder B-Inhalt transportiert werden soll, oder Level C-Inhalt. Das Vorgehen wird im Folgenden für beide Fälle beschrieben:

6.1.1 DATEX II v3 Level A oder B

Bei der Erzeugung der XML-Schemas des Datenprofiles der Publikation werden mehrere Schemas erzeugt. Das in jede Instanz der Publikation einzubettende Schema heißt DATEXII_3_D2Payload.xsd und definiert den Namenraum http://datex2.eu/schema/3/d2Payload. Der Datengeber muss diese Inhaltsdatenschemas zusammen mit der Variante von ExchangeInformation.xsd sowie MessageContainer.xsd für Level A und B in der Benutzeroberfläche zu seiner Publikation hochladen [DATEXIIv3Exc].

6.1.2 DATEX II v3 Level C

Bei der Erzeugung der XML-Schemas des Datenprofiles der Publikation werden mehrere Schemas erzeugt. Das in die Instanzen der Publikation einzubettende Schema heißt LevelC_3_D2Payload.xsd und definiert den Namenraum http://levelC/schema/3/d2Payload. Der Datengeber muss diese Inhaltsdatenschemas zusammen mit der Variante von ExchangeInformation.xsd sowie MessageContainer.xsd für Level C in der Benutzeroberfläche zu seiner Publikation hochladen [DATEXIIv3Exc]

i	**Wichtiger Hinweis** DATEX II v3 Inhalte auf dem MDM müssen immer auf einem von der abstrakten Klasse PayloadPublication im DATEX II v3 Paket Common abgeleiteten XML-Einstiegselement aufbauen. Dies ist unabhängig davon, welche Kompatibilitätsstufe genutzt wird, da der Exchange 2020 MessageContainer ein solches Objekt erwartet. Das aus dem zugehörigen DATEX II Paket Common generierte XML-Schema muss immer Bestandteil der Inhaltsdatenschemas sein, also entweder DATEXII_3_Common.xsd (Level A oder B) oder LevelC_3_Common.xsd (Level C). Benutzer, die eine Level C-Publikation mit anderer Struktur anstreben, müssen entsprechende manuelle Anpassungen auf XML-Schema-Ebene durchführen. Bei Bedarf sollten sie sich für Unterstützung an den MDM-Support wenden.

6.2 SOAP-Schnittstelle

6.2.1 Datengeberseite

6.2.1.1 Client Pull SOAP

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

6.2.1.1.1 Anbieten eines Webservices

Das Datengebersystem muss einen Webservice anbieten, der aufgrund der DATEX II Snapshot Pull WSDL [DATEXIIv3Pull] definiert ist. Als Input wird dabei nichts erwartet, als Output erwartet das MDM-Brokersystem die angeforderten Daten in einem MessageContainer im DATEX II Format gemäß dem Minimal-MessageContainer-Profil im Schema MessageContainer.xsd [DATEXIIv3Exc].

Es liegt in der Verantwortung des Datengebers, das verpflichtende exchangeInformation-Element mit seinen Elementen exchangeContext und dynamicInformation zu definieren. Um standardkonform die Daten bereitzustellen, ist zu beachten, dass das Element codedExchangeProtocol auf den Wert „snapshotPull“ gesetzt werden soll. Unabhängig davon ersetzt der MDM das codedExchangeProtocol-Element mit dem jeweils dem Auslieferungsprotokoll entsprechenden Wert (siehe Kap. 4.1), um standardkonformen Datennehmersystemen eine korrekte Verarbeitung zu ermöglichen.

Über die MDM-Administrations-Komponente muss der Datengeber die URL seines Service-Endpoints in der Publikations-Konfiguration hinterlegen, an dem der MDM das Datenpaket abrufen soll.

6.2.1.1.2 Aufrufen eines Webservices

Das MDM-Brokersystem stellt einen aufgrund der DATEX II Snapshot Pull WSDL [DATEXIIv3Pull] definierten Webservice-Client zum Aufruf von Webservices bereit. Dieser Webservice muss Daten gemäß dem Schema MessageContainer.xsd [DATEXIIv3Exc] zurückliefern.

Für den Fall, dass im Datengebersystem kein Datenpaket für eine Abgabe vorhanden ist, erwartet das MDM-Brokersystem folgende Antwort:

- HTTP Response Code 200 Ok
- ein Minimal MessageContainer-Element ohne Payload

Beispiel-Response:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <con:messageContainer xmlns:con="http://datex2.eu/schema/3/messageContainer"
                          xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                          xsi:schemaLocation="http://datex2.eu/schema/3/
                                  messageContainer/DATEXII_3_MessageContainer.xsd"
                          modelBaseVersion="3">
      <con:exchangeInformation modelBaseVersion="3">
        <ex:exchangeContext>
          <ex:codedExchangeProtocol>snapshotPull</ex:codedExchangeProtocol>
          <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
          <ex:supplierOrCisRequester/>
        </ex:exchangeContext>
        <ex:dynamicInformation>
         <ex:exchangeStatus>online</ex:exchangeStatus>
         <ex:messageGenerationTimestamp>%TIMESTAMP%</ex:messageGenerationTimestamp>
        </ex:dynamicInformation>
      </con:exchangeInformation>
    </con:messageContainer>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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 potenzielle Datennehmer in entsprechenden Paketpuffern zwischengespeichert. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

6.2.1.2 Publisher Push SOAP

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-Brokersystems unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

6.2.1.2.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der aufgrund der Spezifikation DATEX II Snapshot Push WSDL [DATEXIIv3Push] definiert ist. Als Input werden die zu überliefernden Daten in einer MessageContainer-Instanz im Body-Element des SOAP-Envelopes erwartet.

Es liegt in der Verantwortung des Datengebers, das verpflichtende exchangeInformation-Element mit seinen Elementen exchangeContext und dynamicInformation zu definieren. Um standardkonform die Daten bereitzustellen, ist zu beachten, dass das Element codedExchangeProtocol auf den Wert „snapshotPush“ gesetzt werden soll. Unabhängig davon ersetzt der MDM das codedExchangeProtocol-Element mit dem jeweils dem Auslieferungsprotokoll entsprechendem Wert (siehe Kap. 4.1), um standardkonformen Datennehmersystemen eine korrekte Verarbeitung zu ermöglichen.

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://broker.mdm-portal.de/BASt-MDM-Interface/srv/<publication ID>/snapshotPushService
```

Beispiel:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
      <con:messageContainer xmlns:con="http://datex2.eu/schema/3/messageContainer"
                xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
                xmlns:d2="http://datex2.eu/schema/3/d2Payload"
                xmlns:loc="http://datex2.eu/schema/3/locationReferencing"
                xmlns:com="http://datex2.eu/schema/3/common"
                xmlns:sit="http://datex2.eu/schema/3/situation"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://datex2.eu/schema/3/messageContainer
                                    ./DATEXII_3_MessageContainer.xsd"
                modelBaseVersion="3">
        <con:payload lang="en"
               xsi:type="sit:SituationPublication"
               modelBaseVersion="3">
          ...
        </con:payload>
        <con:exchangeInformation modelBaseVersion="3">
          <ex:exchangeContext>
            <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
            <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
```

```
            <ex:supplierOrCisRequester/>
          </ex:exchangeContext>
          <ex:dynamicInformation>
            <ex:exchangeStatus>online</ex:exchangeStatus>
            <ex:messageGenerationTimestamp>2021-07-21T13:00:00
            </ex:messageGenerationTimestamp>
          </ex:dynamicInformation>
        </con:exchangeInformation>
      </con:messageContainer>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.2.1.2.2 Aufrufen des Webservices

Das Datengebersystem muss einen aufgrund der DATEX II Snapshot Push WSDL [DATEXIIv3Push] definierten Webservice Client zum Aufruf des Webservices bereitstellen. Der Webservice muss am Publikationsspezifischen 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.

Eine erfolgreiche Anlieferung beantwortet der MDM mit einer Response in Form einer DATEX II ExchangeInformation mit dem positiven returnStatus „ack“ gemäß der Schemadefinition in ExchangeInformation.xsd [DATEXIIv3Exc].

Beispiel:

```
<ex:putSnapshotDataOutput xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
        xmlns:com="http://datex2.eu/schema/3/common"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://datex2.eu/schema/3/exchangeInformation
                            DATEXII_3_ExchangeInformation.xsd"
        modelBaseVersion="3">
  <ex:exchangeContext>
    <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
    <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
    <ex:supplierOrCisRequester></ex:supplierOrCisRequester>
  </ex:exchangeContext>
  <ex:dynamicInformation>
    <ex:exchangeStatus>online</ex:exchangeStatus>
    <ex:messageGenerationTimestamp>2021-08-
06T15:49:33.600+02:00</ex:messageGenerationTimestamp>
    <ex:returnInformation>
      <ex:returnStatus>ack</ex:returnStatus>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

Eine fehlerhafte Anlieferung beantwortet der MDM hingegen mit einer Response in Form einer DATEX II ExchangeInformation mit dem negativen returnStatus „fail“ gemäß der Schemadefinition in ExchangeInformation.xsd [DATEXIIv3Exc], z. B. wenn die Publikation nicht für das SOAP-Push-Verfahren konfiguriert ist.

Beispiel:

```
<ex:putSnapshotDataOutput xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
        xmlns:com="http://datex2.eu/schema/3/common"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://datex2.eu/schema/3/exchangeInformation
DATEXII_3_ExchangeInformation.xsd"     modelBaseVersion="3">
  <ex:exchangeContext>
    <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
    <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
    <ex:supplierOrCisRequester></ex:supplierOrCisRequester>
  </ex:exchangeContext>
  <ex:dynamicInformation>
    <ex:exchangeStatus>online</ex:exchangeStatus>
    <ex:messageGenerationTimestamp>2021-08-06T15:49:33.600+02:00
    </ex:messageGenerationTimestamp>
    <ex:returnInformation>
      <ex:returnStatus>fail</ex:returnStatus>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

6.2.2 Datennehmerseite

6.2.2.1 Client Pull SOAP

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

6.2.2.1.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice an, der aufgrund der Spezifikation [DATEXIIv3Pull] definiert ist. Als Input wird dabei in der URL die Subskriptions-ID erwartet, als Output bekommt der Datennehmer die angeforderten Daten im payload-Element eines MessageContainers im DATEX II Exchange 2020 Format zurück. Aufgrund der übermittelten Subskriptions-ID kann die MDM-Plattform den zugehörigen Paketpuffer sowie das Datenpaket ermitteln.

Hinweis: Enthält der Paketpuffer zum Zeitpunkt der Anfrage kein Datenpaket beantwortet der MDM die Anfrage mit einem MessageContainer ohne payload-Element. (vgl. 6.2.1.1.2)

6.2.2.1.2 Aufrufen des Webservices

Das Datennehmersystem muss einen aufgrund der Spezifikation [DATEXIIv3Pull] 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://broker.mdm-portal.de/BASt-MDM-Interface/srv/<Subskriptions-ID>/snapshotPull
```

Beispiel:

```
<?xml version='1.0'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
                   xmlns:xsd='http://www.w3.org/2001/XMLSchema'
                   xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
  <SOAP-ENV:Body>
    <con:messageContainer xmlns:con='http://datex2.eu/schema/3/messageContainer'
                xmlns:ex='http://datex2.eu/schema/3/exchangeInformation'
                xmlns:d2='http://datex2.eu/schema/3/d2Payload'
                xmlns:loc='http://datex2.eu/schema/3/locationReferencing'
                xmlns:com='http://datex2.eu/schema/3/common'
                xmlns:sit='http://datex2.eu/schema/3/situation'
                xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
                xsi:schemaLocation='http://datex2.eu/schema/3/messageContainer
                                    ./DATEXII_3_MessageContainer.xsd'
                modelBaseVersion='3'>
      <con:payload lang='en'
             xsi:type='sit:SituationPublication'
             modelBaseVersion='3'>
        ...
      </con:payload>
      <con:exchangeInformation modelBaseVersion='3'>
        <ex:exchangeContext>
          <ex:codedExchangeProtocol>snapshotPull</ex:codedExchangeProtocol>
          <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
          <ex:supplierOrCisRequester/>
        </ex:exchangeContext>
        <ex:dynamicInformation>
          <ex:exchangeStatus>online</ex:exchangeStatus>
          <ex:messageGenerationTimestamp>2021-07-21T13:00:00
          </ex:messageGenerationTimestamp>
        </ex:dynamicInformation>
      </con:exchangeInformation>
    </con:messageContainer>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.2.2.2 Publisher Push SOAP

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.

6.2.2.2.1 Anbieten eines Webservices

Das Datennehmersystem muss einen Webservice anbieten, der aufgrund der Spezifikation [DATEXIIv3Push] definiert ist. Als Input sendet der MDM im body-Element einen MessageContainer mit den angeforderten Daten, als Antwort erwartet die MDM-Plattform eine DATEX II ExchangeInformation mit einem positiven returnStatus „ack“ gemäß der Schemadefinition in ExchangeInformation.xsd [DATEXIIv3Exc].

6.2.2.2.2 Aufrufen des Webservices

Das MDM-Brokersystem stellt einen auf Basis von [DATEXIIv3Push] 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:

```
<ex:putSnapshotDataOutput xmlns:ex="http://datex2.eu/schema/3/exchangeInformation"
        xmlns:com="http://datex2.eu/schema/3/common"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://datex2.eu/schema/3/
                            exchangeInformation DATEXII_3_ExchangeInformation.xsd"
        modelBaseVersion="3">
  <ex:exchangeContext>
    <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
    <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
    <ex:supplierOrCisRequester></ex:supplierOrCisRequester>
  </ex:exchangeContext>
  <ex:dynamicInformation>
    <ex:exchangeStatus>online</ex:exchangeStatus>
    <ex:messageGenerationTimestamp>2021-08-06T15:49:33.600+02:00
    </ex:messageGenerationTimestamp>
    <ex:returnInformation>
      <ex:returnStatus>ack</ex:returnStatus>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

6.3 HTTPS-Schnittstelle

6.3.1 Datengeberseite

6.3.1.1 Client Pull HTTPS

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 die Regeln des Snapshot Pull aus dem [DATEXIIv3Annex], *Anhang C - "Snapshot Pull with simple http server" profile definition*.

DATEX II vDabei ist zu berücksichtigen, dass die weiteren, optionalen Regeln keine Anwendung finden. Die Optionen zur Authentisierung ([DATEXIIv3Annex], *Anhang C - "Snapshot Pull with simple http server" profile definition, Authentication*) finden keine Anwendung, da sie bei der Verwendung des für den MDM verpflichtenden HTTPS-Verfahrens obsolet sind. Siehe hierzu auch Anhang B – DATEX II HTTP Protokollunterstützung.

6.3.1.1.1 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.

Beachten Sie auch die Hinweise in Kapitel 4.5, Nutzung des „If-Modified-Since" Header-Feldes.

6.3.1.1.2 Response an die MDM-Plattform

Das Datengebersystem muss nach Erhalt des Requests eine HTTPS Response erzeugen, deren Message-Body aus den angeforderten DATEX II v3 Daten besteht. Hier wird ein MessageContainer-Objekt erwartet, das dem Minimalprofil der MessageContainer.xsd [DATEXIIv3Exc] genügt. Gemäß [DATEXIIv3Annex] *Anhang C - "Snapshot Pull with simple http server" profile definition, Basic request / response pattern*, hat die Response im Content-Type „text/xml; charset=utf-8" vorzuliegen und kann GZIP-codiert angeliefert werden.

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

Beispiel:

```
<?xml version='1.0'?>
<con:messageContainer xmlns:con='http://datex2.eu/schema/3/messageContainer'
            xmlns:ex='http://datex2.eu/schema/3/exchangeInformation'
            xmlns:d2='http://datex2.eu/schema/3/d2Payload'
            xmlns:loc='http://datex2.eu/schema/3/locationReferencing'
            xmlns:com='http://datex2.eu/schema/3/common'
            xmlns:sit='http://datex2.eu/schema/3/situation'
```

```
            xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
            xsi:schemaLocation='http://datex2.eu/schema/3/messageContainer
                                ./DATEXII_3_MessageContainer.xsd'
            modelBaseVersion='3'>
 <con:payload lang='en'
         xsi:type='sit:SituationPublication'
         modelBaseVersion='3'>
    ...
  </con:payload>
  <con:exchangeInformation modelBaseVersion='3'>
    <ex:exchangeContext>
      <ex:codedExchangeProtocol>snapshotPull</ex:codedExchangeProtocol>
      <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
      <ex:supplierOrCisRequester/>
    </ex:exchangeContext>
    <ex:dynamicInformation>
      <ex:exchangeStatus>online</ex:exchangeStatus>
      <ex:messageGenerationTimestamp>2021-07-21T13:00:00
      </ex:messageGenerationTimestamp>
    </ex:dynamicInformation>
  </con:exchangeInformation>
</con:messageContainer>
```

6.3.2 Datennehmerseite

6.3.2.1 Client Pull HTTPS

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

6.3.2.1.1 Request an die MDM-Plattform

Das Datennehmersystem soll einen HTTPS GET-Request an die MDM-Plattform schicken. Aufgrund der Subskriptions-ID ist der zugehörige Paketpuffer sowie das Datenpaket festgelegt. Alternativ kann ein HTTPS POST-Request genutzt werden.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://broker.mdm-portal.de/BASt-MDM-
Interface/datexv3/http/content.xml?subscriptionID=<Subskriptions-ID>
```

Beachten Sie auch die Hinweise in Kapitel 4.5, Nutzung des „If-Modified-Since“ Header-Feldes.

6.3.2.1.2 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 Snapshot Pull Profil ([DATEXIIv2PSM], Anhang C – "Snapshot Pull with simple http server" profile definition, Overall *presentation*) wird der Inhalt immer mit einer MessageContainer-Instanz ausgeliefert. Zusätzlich hat die Response den Content Type „text/xml; charset=utf-8" und wird – abweichend zum Standard – ausschließlich GZIP-komprimiert verschickt. **Anfragen mit dem „identity encoding", oder andere Komprimierungsformate werden mit dem Fehlercode HTTP 406 (Not Acceptable) quittiert.**

Als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 10 beschriebenen Bedeutungen gelten:

	Beschreibung
Request	Request GET /BASt-MDM-Interface/datexv3/http/content.xml?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx < messageContainer > … </messageContainer>
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: - 304: Not Modified, die angeforderte Ressource wird nicht erneut übertragen - 401: Authentifizierungsfehler - 204: Kein Datenpaket im Paketpuffer zur Subskription - 404: Keine Subskription oder eine nicht mehr gültige Subskription zum Subskriptionsparameter gefunden - 406: Ressource wurde nicht im Format GZIP angefordert - 503: Service Unavailable (z. B. bei Wartung)

Tabelle 10: Request/Response zwischen MDM-Plattform/Datennehmersystem beim Client Pull HTTPS

7 Container

7.1 SOAP-Schnittstelle

7.1.1 Datengeberseite

7.1.1.1 Client Pull SOAP

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.

7.1.1.1.1 Anbieten eines Webservices

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

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

7.1.1.1.2 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.

7.1.1.2 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-Brokersystems unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

7.1.1.2.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice mit der Methode pushContainerData 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. Es wird jeweils ein Objekt vom Typ containerdata erwartet.

Beispiel:

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

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

7.1.2 Datennehmerseite

7.1.2.1 Client Pull SOAP

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

7.1.2.1.1 Anbieten eines Webservices

Das MDM-Brokersystem bietet einen Webservice mit der Methode pullContainerDataClient an, der als Input eine Subskriptions-ID (Typ subscriptionId) in den XML-Daten und einen optionalen Zeitstempel (Typ timestamp - beinhaltet den Erstellungszeitpunkt der Anfrage) erwartet. Als Output werden die Daten im Containerformat (Typ containerdata) zurückgeliefert.

Beispiel:

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

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

7.1.2.2 Publisher Push SOAP

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.

7.1.2.2.1 Anbieten eines Webservices

Das Datennehmersystem muss einen Webservice mit der Methode pushContainerData anbieten, der aufgrund der Containerformat Spezifikation [MCS] definiert ist. Als Input muss ein Datenpaket vom Typ des Containerformates (Typ containerdata) akzeptiert werden und als Output ist eine Statusnachricht (ebenfalls vom Typ containerdata) zu liefern.

7.1.2.2.2 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.

7.2 HTTPS-Schnittstelle

7.2.1 Datengeberseite

7.2.1.1 Client Pull HTTPS

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.

7.2.1.1.1 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:

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

7.2.1.1.2 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 11 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 11: Request/Response zwischen Datengebersystem/MDM-Plattform beim Client Pull HTTPS

Beispiel:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<container xmlns="https://ws.bast.de/container/TrafficDataService"
           xmlns:ns2="https://schemas.xmlsoap.org/ws/2002/07/utility"
           xmlns:ns3="https://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>
```

7.2.1.2 Publisher Push HTTPS

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

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

Beispiel:

```
<?xml version='1.0' encoding='UTF-8'?>
<ns3:containerRootElementEl xmlns="https://www.w3.org/2000/09/xmldsig#"
                        xmlns:ns2="https://schemas.xmlsoap.org/ws/2002/07/utility"
                    xmlns:ns3="https://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>
```

7.2.1.2.2 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 12 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 12: Request/Response zwischen Datengebersystem/MDM-Plattform beim Publisher Push HTTPS

7.2.2 Datennehmerseite

7.2.2.1 Client Pull HTTPS

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.

7.2.2.1.1 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://broker.mdm-portal.de/BASt-MDM-
Interface/srv/container/v1.0?subscriptionID=<Subskriptions-ID>
```

7.2.2.1.2 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 13 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 13: Request/Response zwischen MDM-Plattform/Datennehmersystem beim Client Pull HTTPS

7.2.2.2 Publisher Push HTTPS

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

7.2.2.2.1 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.

7.2.2.2.2 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 14 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 14: Request/Response zwischen MDM-Brokersystem/Datennehmersystem beim Publisher Push HTTPS

8 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.

8.1 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.

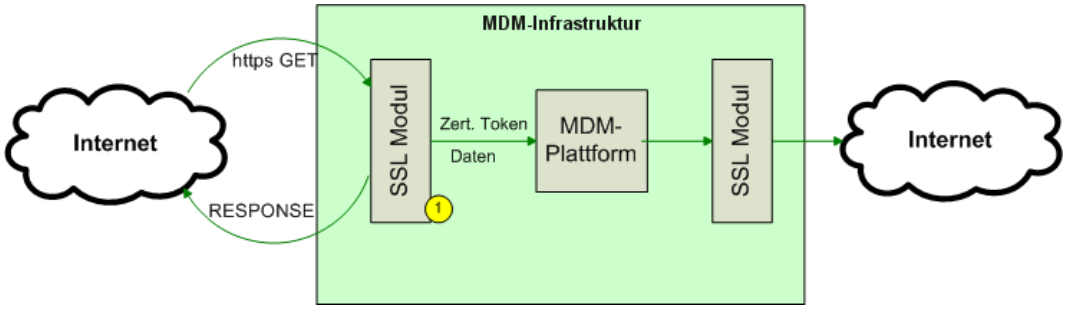

Abbildung 4: Übersicht Sicherheitsarchitektur

Das SSL-Modul 1 in Abbildung 4 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.

8.2 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.

8.3 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.

8.4 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:

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

https://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 9

8.5 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.

8.6 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

9 Anhang A- p12-Datei für Apache Server Konfiguration aufbereiten

Die Apache-Serverkonfiguration kann keine Dateien des Typen 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:

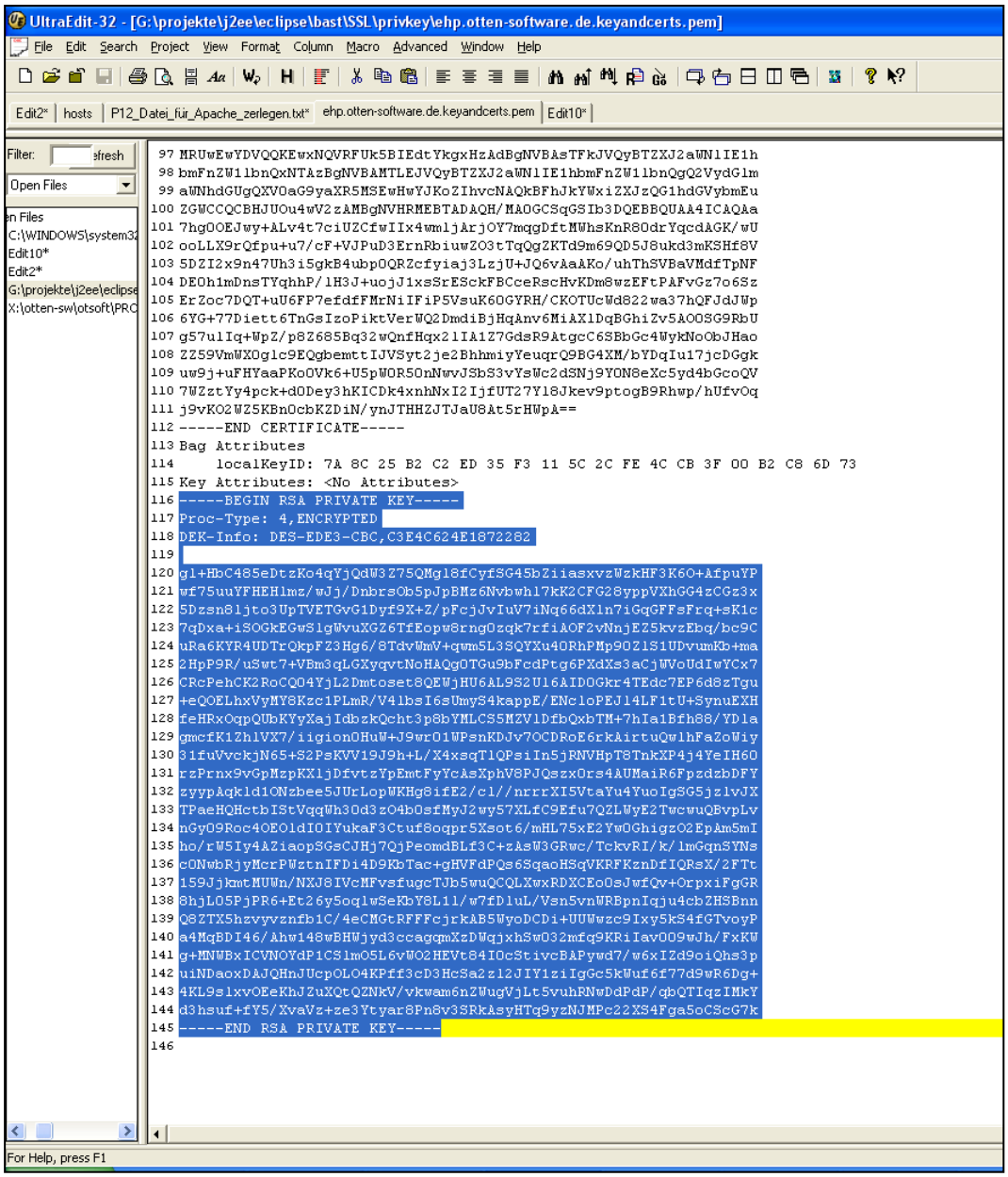

Abbildung 5: Datei <sammeldatei.pem>

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:

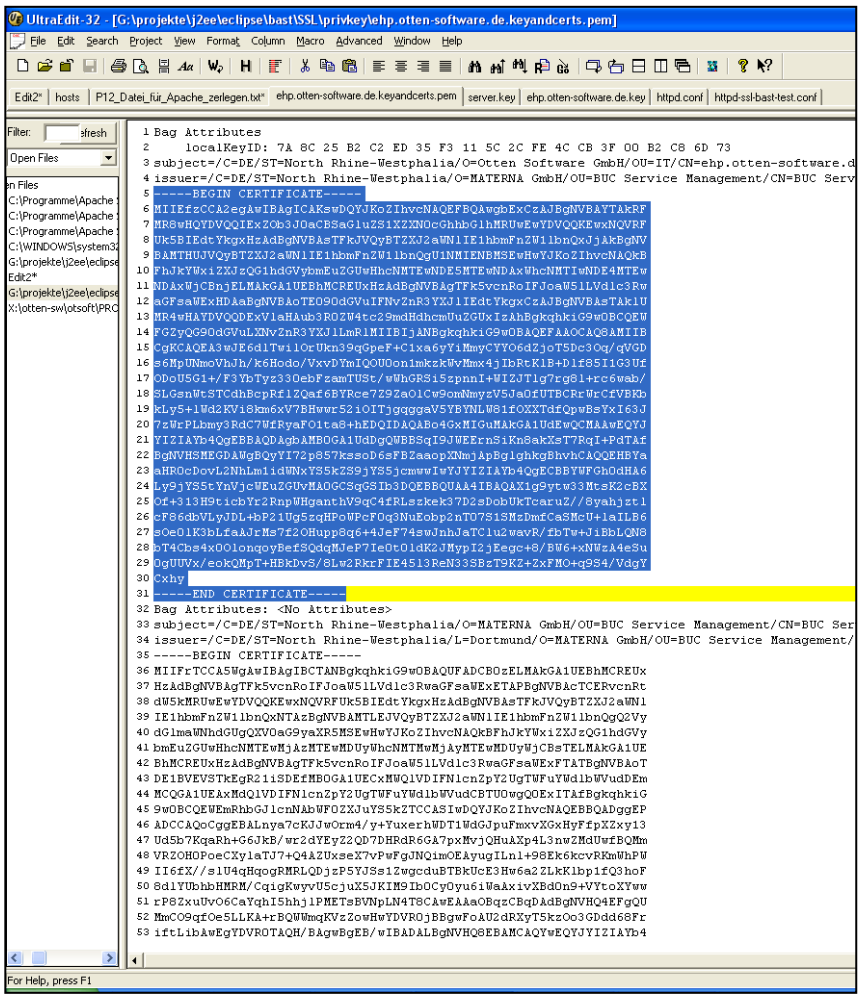

Abbildung 6: Datei <sammeldatei.pem>

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"
```

10 Anhang B – DATEX II HTTP Protokollunterstützung

Regel	Verweis auf Regel in [DATEXIIv2PSM], Kapitel 4	Verweis auf Regel in [DATEXIIv3Annex]	DATEX II v2	DATEX II v3
Suppliers and Clients SHALL use the HTTP/1.1 protocol. Clients and Suppliers shall fully comply with the HTTP/1.1 protocol specification in RFC 2616, as of June 1999.	C.1	Basic request / response pattern: 1.	✓	✓
Clients SHALL use the HTTP GET or POST method of the HTTP REQUEST message to request data from the Supplier.	C.2	Basic request / response pattern: 2.	✓	✓
Suppliers SHALL use an HTTP RESPONSE message to respond to requests.	C.3	Basic request / response pattern: 3.	✓	✓
Suppliers SHALL NOT respond to HTTP REQUEST messages using the GET or POST methods by responding with 405 (Method Not Allowed) or 501 (Not Implemented) return codes.	C.4	Basic request / response pattern: 4.	✓	✓
Suppliers Shall set the ‘Last-Modified’ header field in HTTP RESPONSE messages that provide payload data (response code 200) to the value that the information product behind the URL was last updated.	C.5	Basic request / response pattern 5.	✓	✓
Clients SHOULD set the ‘If-Modified-Since’ header field in all HTTP REQUEST messages if they already hold a consistent set of data from a particular URL in their database and the last modification time of that data is known from the ‘Last-Modified’ header field of the HTTP header of the HTTP RESPONSE message within which the payload data was received.	C.6	Basic request / response pattern: 6.	✓	✓

Regel	Verweis auf Regel in [DATEXIIv2PSM], Kapitel 4	Verweis auf Regel in [DATEXIIv3Annex]	DATEX II v2	DATEX II v3
When setting the 'If-Modified-Since' header field, the Client SHALL copy the value of the Last-Modified header field received within the last successful HTTP RESPONSE containing payload (response code 200) message into this field.	C.7	Basic request / response pattern: 7.	✓	✓
Suppliers SHOULD provide XML coded DATEX II payload as "text/xml" media type. Suppliers SHOULD state the used character set via the "charset" parameter; Suppliers SHOULD use the UTF-8 character set, i.e., the "Content-Type" response-header field SHOULD state "text/xml; charset=utf-8.	C.8	Basic request / response pattern: 8.	✓	✓
Clients MUST accept "identity" content-coding; Clients SHOULD (and if they do, prefer to) accept "gzip" content-coding; Clients MAY accept other "content-coding" values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry1 as long as they also accept "identity" and "gzip" content-coding.	C.9	Basic request / response pattern: 9.	✓	✓
When including an "Accept-Encoding" request-header field in an HTTP REQUEST message, the Client MUST NOT exclude acceptance of "identity" content-coding.	C.10	Basic request / response pattern: 10.	✓	✓
Suppliers MUST provide "identity" content-coding of the payload; Suppliers SHOULD provide "gzip" content-coding of the payload; Suppliers MAY provide other "content-coding" values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry as long as they also provide "identity" and "gzip" content-coding	C.11	Basic request / response pattern: 11.	✗	✗

Regel	Verweis auf Regel in [DATEXIIv2PSM], Kapitel 4	Verweis auf Regel in [DATEXIIv3Annex]	DATEX II v2	DATEX II v3
Clients SHOULD fill access credentials they MAY have received during the subscription negotiation process into the ‘Authorization’ header field of the HTTP REQUEST message.	C.13	Authentication	✗	✗
Server providing access credentials (user name & password) during the subscription negotiation phase MAY respond with response code 401 (Unauthorized) to HTTP REQUESTS that do not contain valid access credentials in the ‘Authorization’ header field.	C.14		✗	
Servers SHALL produce and Clients SHALL process the following return codes: • 200 (OK), in responses carrying payload, • 304 (Not Modified), if no payload is send because of the specification in the ‘If-Modified-Since’ header, • 503 (Service Unavailable), if an active HTTP server is disconnected from the content feed, • 404 (Not Found), if a file based HTTP server does not have a proper payload document stored in the place associated to the URL.	C.15	Additional Rules	(✓) Abweichung: 403 anstatt 401	✓ Zusätzlich 204 bei leerem Paketpuffer
Payload data for Information products SHALL be denoted by a URL according to the following convention: d2lcp_infop = "http://" host [":" port] infop_path "/content.xml" ["?" query] where “infop_path” is a “path” component as specified in section 3.3 of [RFC 2396], but excluding the last path segment.	C.16	Describing payload and interfaces	✗	✓