mobil:thek

Deutschlands Plattform für Daten,
die etwas bewegen.

Technische Schnittstellenbeschreibung

Version 1.2.1 – 04.03.2024

Bundesministerium
für Digitales
und Verkehr

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Einleitung

Die Mobilithek 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 die Mobilithek 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 der Mobilithek vorausgesetzt.

Die Datenübertragung zwischen der Mobilithek 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 Kapitel gegliedert:

- Kapitel 1 enthält eine kurze Übersicht, die referenzierten Dokumente sowie das Abkürzungsverzeichnis.
- Im Kapitel 2 werden die Komponenten der Mobilithek vorgestellt.
- Kapitel 3 behandelt die verfügbaren Datenformate.
- Die Schnittstellen der Mobilithek für die M2M-Kommunikation werden in Kapitel 4 grundlegend beschrieben.
- Kapitel 5 beschreibt Schnittstellen, die vom Datenformat unabhängig sind.
- Kapitel 6 beschreibt im Detail das Format DATEX II v2.
- Kapitel 7 beschreibt entsprechend DATEX II v3.
- Kapitel 8 beschreibt das Container-Format der Mobilithek.
- Kapitel 9 beschreibt die Handhabung anderer Datenformate.
- Kapitel 10 beschreibt die Maßnahmen, mit denen die M2M-Kommunikation abgesichert wird.

1.3 Referenzierte Dokumente

1.3.1 Allgemein

[Quelle]	Herausgeber
[FAQ]	"Frequently asked Questions" https://mobilithek.info/help/FAQ
[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]	Containerformat Spezifikation Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie Dokumentation; siehe Container Spezifikation V1.1 – 02|2014

[Quelle]	Herausgeber
[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 Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie WSDL; siehe Mobilithek: OCIT-C 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/
[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: https://docs.datex2.eu/_static/data/v2.0/DATEXII_-_ExchangePSM_0.pdf
[DATEXIIv2Pull]	DATEX II v2.0 Pull wsdl Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie WSDL; siehe Mobilithek - Pull: DATEXII v2
[DATEXIIv2Push]	DATEX II v2.0 Push wsdl Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie WSDL; siehe Mobilithek - Push: DATEXII v2
[DATEXIIv2Schema]	DATEX II XML Schema 2.3: https://docs.datex2.eu/_static/data/v2.3/DATEXIISchema_2_2_3_0.zip
[DATEXIIv2SDG]	DATEX II v2.3 Software Developers Guide: https://docs.datex2.eu/_static/data/v2.3/DATEXII-DevGuide.pdf
[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: https://docs.datex2.eu/_static/data/v2.3/DATEXII-UserGuide.pdf

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 Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie WSDL; siehe Mobilithek - Pull: DATEXII v3
[DATEXIIv3Push]	DATEX II v3.0 Snapshot Push wsdl Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie WSDL; siehe Mobilithek - Push: DATEXII v3
[DATEXIIv3Exc]	Download für Level A- und Level B-Publikationen: Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie DATEXII V3 Download für Level C-Publikationen: Im Download Bereich (https://mobilithek.info/help/download) in der Kategorie DATEXII V3
[DATEXIIv3ExcUG]	Exchange 2020 User Guide: https://docs.datex2.eu/exchange/2020/userguide/
[DATEXIIv3Spec]	Umfasst die folgenden Dokumente, die auf https://www.datex2.eu bzw. auf der Mobilithek allen Benutzern zum Download bereitstehen: [DATEXIIv3ExcUG], [DATEXIIv3Annex], [DATEXIIv3Pull], [DATEXIIv3Push], [DATEXIIv3Exc]

Tabelle 3: Referenzierte Dokumente (DATEX II v3)

1.4 Abkürzungsverzeichnis

Abkürzung	Auflösung
BASt	Bundesanstalt für Straßenwesen
GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
M2M	Machine-to-Machine
MDM	Mobilitätsdatenmarktplatz
MDV	Metadatenverzeichnis
OCIT	Open Communication Interface for Road Traffic Control Systems
PKI	Public Key Infrastructure

Abkürzung	Auflösung
PSM	Platform Specific Model
RFC	Request for Comments
SDG	Software Developers Guide
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator
UTF	UCS Transformation Format
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

Tabelle 4: Abkürzungsverzeichnis

2 Komponenten der Mobilithek im Überblick

Die Mobilithek setzt sich aus vier Komponenten zusammen, die jeweils unterschiedliche Aufgaben übernehmen.

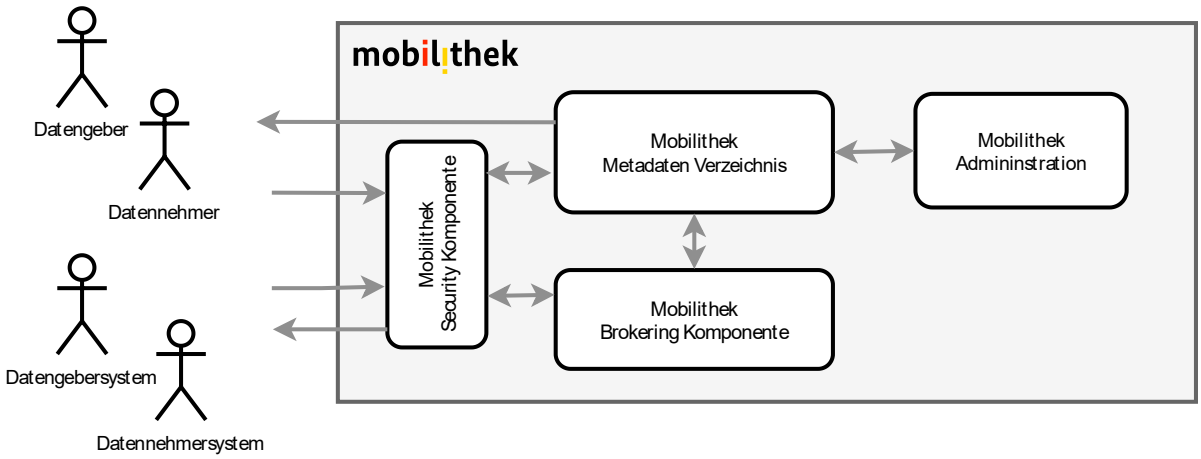

Abbildung 1: Komponenten der Mobilithek

Komponente	Beschreibung
Securitykomponente	Über die Securitykomponente können Datennehmersystem/Datengebersystem oder Datennehmer/Datengeber authentisiert werden, um Dienste nutzen zu können.
Metadatenverzeichnis	Das Metadatenverzeichnis dient zur Verwaltung aller Informationen hinsichtlich der in der Mobilithek bereitgestellten Publikationen.
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 [FAQ].

Tabelle 5: Übersicht der Komponenten der Mobilithek

Durch die Mobilithek 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 Mobilithek 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 Mobilithek unterstützt das auf XML oder JSON (DATEX II V3) 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 größtmögliche Kompatibilität mit dem MDM zu erreichen, wird weiterhin das von konkreten Formaten unabhängige, mit dem MDM eingeführte, Containerformat unterstützt. Über dieses Format können beliebige XML- und Binärdaten übertragen werden.
- Des Weiteren können über die HTTPS Schnittstelle prinzipiell beliebige "Payloads" im HTTP Body übertragen werden, ohne die Notwendigkeit diese in das Containerformat vorher zu verpacken.

Bei der Anlieferung eines Datenpakets an der Brokerschnittstelle der Mobilithek wird die Validität der Daten hinsichtlich Konformität zu einem hinterlegten Dateischema überprüft. Ebenso wird eine Prüfung auf Schadsoftware durchgeführt. Der System Administrator der Mobilithek kann diese Prüfungen für einzelne Publikationen ausschalten.

Die Web-GUI der Mobilithek erlaubt über eine Schaltfläche das jeweils aktuell im Mobilithek gespeicherte Datenpaket herunterzuladen.

Wird eine Validierung des Datenpakets durchgeführt, wird das 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. Für Datenpakete im XML Format ist das Schema über die URL im „schemaLocation“-Attribut zu referenzieren, um Datennehmern eine automatische Validierung der Pakete mit den gleichen Voraussetzungen zu ermöglichen. Bei Datenpaketen im JSON Format wird das Schema im $schema-Attribut referenziert. Sofern dieses weitere Subschemas inkludiert, sind diese über das $ref-Attribut zu spezifizieren. Die referenzierten Subschemas müssen dabei immer mit ihrem gesamten Namen inklusive dem Dateinamensuffix angegeben werden: *"$ref": "<subschema Dateiname>"*. Datengebern wird empfohlen, in den Datenpaketen, die Schemas zu spezifizieren, die für die Publikation in der Mobilithek hinterlegt sind und diese auch für eigenen Validierungen zu benutzen. Die URLs, über die die Schemas referenziert werden können, sind über das GUI in den Publikations-Details ermittelbar. Damit kann erreicht werden, dass alle am Datenaustausch beteiligten Parteien (Datengeber, Mobilithek und Datennehmer) die gleichen Schemas für Validierungen benutzen.

Die Mobilithek akzeptiert ein Datenpaket unabhängig von deren Validität oder dem Ergebnis der Prüfung auf Schadsoftware und liefert dieses auch an die Datennehmer aus, wenn es nicht gegenüber den hinterlegten Schemata valide ist oder der Verdacht auf enthaltene Schadsoftware besteht.

3.1 DATEX II

DATEX II ist ein europaweiter Standard zum Austauschen von Mobilitätsdaten. Für dieses Kapitel werden grundlegende Kenntnisse der DATEX II Spezifikation vorausgesetzt [DATEXIIv2Spec] bzw. [DATEXIIv3Spec]. Die Mobilithek unterstützt sowohl DATEX II v2 als auch v3.

DATEX II definiert XML-Strukturen für den Austausch von Mobilitätsdaten. Für DATEX II V3 wird neben XML auch JSON als Austauschformat unterstützt. Die zugrunde liegenden Schemadateien können von der DATEX II Webseite https://www.datex2.eu/ bezogen 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] bzw. [DATEXIIv3Spec] aufgeführt. Der Aufbau der DATEX II Nutzdaten ist für die Mobilithek 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. Die Mobilithek unterstützt grundsätzlich die Verarbeitung von "Delta" Einlieferungen (siehe Kapitel 4.3). Bezüglich DATEX II wird diese Option derzeit nur für v3 unter Verwendung von XML unterstützt. Für DATEX II v2 und DATEX II v3 (JSON) müssen Datengebersystem immer inhaltlich vollständige Datenpakete bereitstellen, die durch die Mobilithek auch unverändert weitergeleitet werden.

3.2 Containerformat

Zusätzlich zu dem im vorherigen Kapitel genannten DATEX II Standard wird durch die Mobilithek weiterhin das mit dem MDM eingeführte "Containerformat" unterstützen. Dieses Format basiert auf XML zur Übermittlung von Daten. Es wurde eigens für den Datenaustausch über den MDM geschaffen und wird von der Mobilithek weiterhin unterstützt. 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.

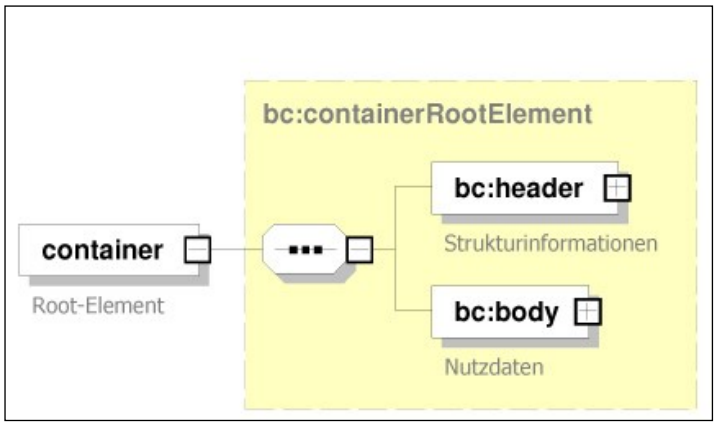

Abbildung 2: Übersicht Containerformat

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.

3.3 Andere Datenformate

Da die Mobilithek die übermittelte "Payload" unverändert weitergibt und auch keinerlei inhaltliche Prüfungen oder Auswertungen vornimmt, ist es auch möglich über die HTTP/REST Schnittstelle der Mobilithek im HTTP Body beliebige Datenformate zu übertragen. Die im HTTP Body enthaltene Payload wird unverändert an Datennehmer weitergegeben.

4 Schnittstellen des Brokersystems der Mobilithek

Das Brokersystem der Mobilithek 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.

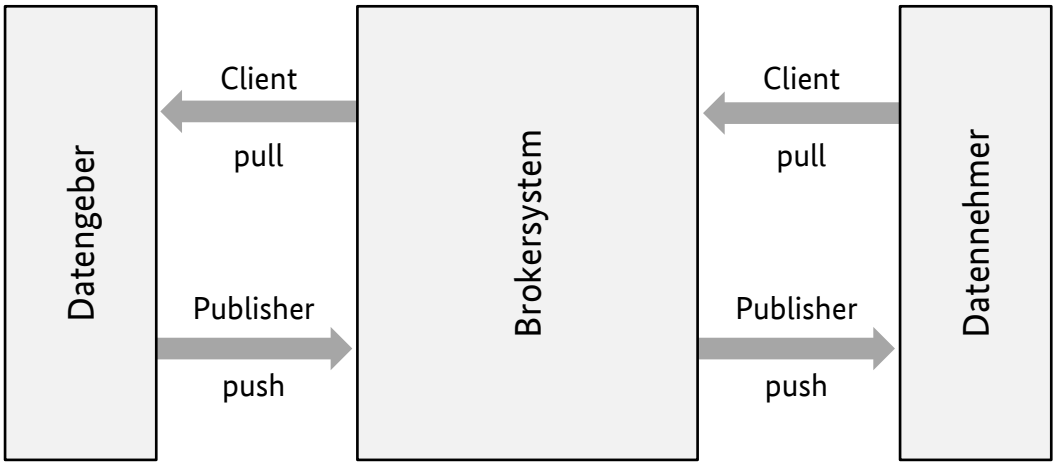

Abbildung 3: Schnittstellen zwischen Datengeber, Brokersystem und Datennehmer

Die Datenpakete, die vom Brokersystem empfangen oder gesendet werden, müssen im DATEX II, im Containerformat oder als "beliebige" Payload 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 der Mobilithek als Client oder als Server auftritt. Client bedeutet hier, dass das System Anfragen an die Mobilithek stellt bzw. aktiv die Verbindung zu dieser aufbaut.

Server bedeutet demgegenüber, dass das System von der Mobilithek angesprochen wird und deren Anfragen beantworten muss. In diesem Fall muss ein Netzwerkzugriff auf das anzubindende System von außen (durch die Mobilithek) erlaubt sein.

		Datengebersystem			**Datennehmersystem**		
		SOAP/ HTTPS	HTTPS	OCIT/ SOAP/ HTTPS	SOAP/ HTTPS	HTTPS	OCIT/ SOAP/ HTTPS
DATEX II v2	Client Pull	6.1.1.1 Server	6.2.1.1 Server	-	6.1.2.1 Client	6.2.2.1 Client	6.3.3 Client
	Publisher Push	6.1.1.2 Client	-	6.3.2 Client	6.1.2.2 Server	-	-
DATEX II v3	Client Pull	7.2.1.1 Server	7.3.1.1 Server	-	7.2.2.1 Client	7.3.2.1 Client	-
	Publisher Push	7.2.1.2 Client	-	-	7.2.2.2 Server	-	-
Container	Client Pull	8.1.1.1 Server	8.2.1.1 Server	-	8.1.2.1 Client	8.2.2.1 Client	-
	Publisher Push	8.1.1.2 Client	8.2.1.2 Client	-	8.1.2.2 Server	8.2.2.2 Server	-
Andere	Client Pull	-	9.1.1 Server	-	-	9.2.1 Client	-
	Publisher Push	-	9.1.2 Client	-	-	9.2.2 Server	-

Tabelle 6: Übersicht über die Schnittstellen des Brokersystems der Mobilithek

Grundsätzlich stehen Datenpakete für Datennehmer nur während der in der Publikationsdefinition hinterlegten Gültigkeit zur Abholung zur Verfügung.

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.

Wichtige Hinweise zu den SOAP Protokoll Endpunkten

1. Für die SOAP Endpunkte veröffentlicht die Mobilithek die dazugehörigen WSDLs, die die Endpunkte spezifizieren. Aus Gründen der Kompatibilität zum MDM nimmt die Mobilithek grundsätzlich auch Datenpakete an, wenn diese nicht konform zum Payload Schema sind, das in der WSDL spezifiziert ist. D.h. die Mobilithek führt keine formale Validierung gegen die WSDL durch. Vielmehr beschränkt sie sich bei der formalen Validierung der Requests auf grundlegende Anforderungen, wie z.B. die Bereitstellung eines wohlgeformten XMLs und eines validen SOAP Requests (SOAP Envelope und SOAP Body ist enthalten) sowie weiterer minimaler Anforderungen, um einen Request erfolgreich verarbeiten zu können. Die spezifischen Anforderungen inklusive des Fehlercodes sind in den jeweiligen protokoll-spezifischen Kapiteln explizit spezifiziert.
2. Als Konsequenz aus Punkt 1 können sich Datennehmer nicht darauf verlassen, dass das ausgelieferte Datenpaket und damit der SOAP Body konform zur WSDL ist.

4.1 Verschlüsselung der Kommunikation

Über die von der Mobilithek 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 Mobilithek ausgestellt werden.

In den Fällen, in denen die Mobilithek als Client agiert nutzt sie ein X.509 konformes Zertifikat um sich beim als Server agierenden Datengeber- oder Datennehmersystem - falls erforderlich - zu authentisieren, siehe Kapitel 10.4.

4.2 Komprimierung

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

Die Datenübermittlung zwischen Mobilithek und Datennehmersystemen findet – abweichend von [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 Unterstützung von "Delta" Datenpaketen

Mobilithek unterstützt grundsätzlich die Verarbeitung von Delta-Datenpaketen.

Empfängt die Mobilithek ein vollständiges Datenpaket so ersetzt dieses alle derzeit im zugehörigen Datenpaketpuffer befindlichen Datenpakete. Ein empfangenes "Delta" Datenpaket speichert die Mobilithek in der Reihenfolge ihres Empfangs im zugehörigen Datenpaketpuffer.

Datennehmersysteme können über die Datennehmer-Pull-Schnittstelle auf alle existierenden Datenpakete inkl. der letzten vollständigen Lieferung in der Reihenfolge der Anlieferung zugreifen. Dieser Mechanismus kann auch von Datennehmern genutzt werden, die als Auslieferungsmodus PUSH konfiguriert haben.

Grundsätzlich können dadurch auch Datennehmer, die nur mittels PULL auf Datenangebote zugreifen, von der Bandbreiten Reduktion profitieren, die durch Nutzung von Delta-Lieferungen entstehen.

Da die Mobilithek den Paketinhalt nicht auswertet, in keinerlei Hinsicht verändert und auch nicht zusammenfügt, ist es in der Verantwortung des Datengebersystems, die Datenpakete in der Reihenfolge in die Mobilithek einzuliefern, die es einem Datennehmersystems ermöglicht, diese zu einer vollständigen Publikation zusammenzusetzen.

Die Kennzeichnung, ob es sich um ein vollständiges oder ein Delta-Datenpaket handelt, ist abhängig vom Datenformat. Aktuell wird Delta-Anlieferung für DATEX II v3 unterstützt.

4.4 Gültigkeitsdauer von Datenpaketen

Über die Benutzeroberfläche kann ein Datengeber optional für eine Publikation eine Gültigkeitsdauer spezifizieren. Sofern eine Gültigkeitsdauer spezifiziert wird, definiert diese die maximale Zeitspanne zwischen der Einlieferung von Datenpaketen für die betreffende Publikation. Wird diese Zeitspanne überschritten, wird der zugehörige Paketpuffer als "veraltet" markiert.

Dies hat im Einzelnen folgende Konsequenzen:

- Der Paketpuffer wird geleert.
- Es werden keine weiteren Datenpakete für dieses Datenangebot an Datennehmer übertragen. Laufende Datenübertragungen werden nicht abgebrochen.
- Bei aktivierter Delta-Unterstützung werden Delta-Pakete verworfen und nicht an Datennehmer weitergeleitet.
- Aus Datennehmersicht stellt sich der Zustand so dar, als seien keine Datenpakete für dieses Datenangebot vorhanden.
- Der Zustand "veraltet" wird aufgehoben, wenn erneut ein vollständiges Datenpaket vom Datengeber bereitgestellt wird.

4.5 Unveränderlichkeitsversprechen

Die Mobilithek 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.

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). Werden Datenpakete eingeliefert, bei denen sich Namespace-Deklarationen im SOAP-Envelope befinden, so werden davon tatsächlich benötigte Deklarationen bei Auslieferung der Dateien in die jeweiligen XML Elemente verschoben. Des Weiteren kann es auch zu einer technisch bedingten Umsortierung von Namespace-Deklarationen kommen. Diese Modifikationen verändern den Dateninhalt inhaltlich und semantisch nicht.

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

4.5.1 DATEX II v2

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <d2LogicalModel xmlns="https://datex2.eu/schema/2/2_0" modelBaseVersion="2">
      <exchange>
        <supplierIdentification>
          <country>de</country>
          <nationalIdentifier>DE-Mobilithek-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>
  </soapenv:Body>
</soapenv:Envelope>
```

4.5.2 DATEX II v3

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv: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="<URL des Schema Files>"
           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>
</soapenv:Body>
```

Beispiel für den Fall, dass die Namespace Deklarationen sich bei Einlieferung am SOAP-Envelope befanden und nun direkt den jeweiligen XML Elementen zugeordnet sind:

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
    <soapenv:Body>
      <con:messageContainer xmlns:con=http://datex2.eu/schema/3/messageContainer>
        <con:payload lang="en"
 modelBaseVersion="3" xmlns:xsi=http://www.w3.org/2001/XMLSchema-
instance xsi:type="sit:SituationPublication">
          <test>Daten</test>
        </con:payload>
          <con:exchangeInformation modelBaseVersion="3">
```

```
            <ex:exchangeContext xmlns:ex=http://datex2.eu/schema/3/exchangeInformation>
              <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
              <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
              <ex:supplierOrCisRequester/>
            </ex:exchangeContext>
          <ex:dynamicInformation xmlns:ex=http://datex2.eu/schema/3/exchangeInformation>
            <ex:exchangeStatus>online</ex:exchangeStatus>
            <ex:messageGenerationTimestamp>
               2022-11-07T14:36:00
            </ex:messageGenerationTimestamp>
          </ex:dynamicInformation>
          </con:exchangeInformation>
    </con:messageContainer>
  </soapenv:Body>
</soapenv:Envelope>
```

Wichtige Hinweise zu DATEX II v3

1. Das Element <codedExchangeProtocol> (erforderliches Element), ist abhängig vom verwendeten Ein/Auslieferungs-Protokoll sowie von der Art des Datenpakets.
2. Bei der Einlieferung eines Datenpakets in die Mobilithek wird folgender Wert erwartet:
    a. "snapshotPush", wenn es sich um ein vollständiges Datenpaket handelt und der Datengeber das Paket mittels Push einliefert.
    b. "deltaPush", wenn es sich um ein Delta-Datenpaket handelt und der Datengeber das Paket mittels Push einliefert.
    c. "snapshotPull", wenn die Mobilithek das Datenpaket vom Datengebersystem abholt und es sich um ein vollständiges Datenpaket handelt.
    d. "deltaPull", wenn die Mobilithek das Datenpaket vom Datengebersystem abholt und es sich um ein Delta-Datenpaket handelt.
3. Bei jeder Auslieferung der Mobilithek an das Datennehmersystem wird der Wert dieses Elements wie folgt gesetzt:
    a. „snapshotPush“, wenn die Mobilithek das Datenpaket an den Datennehmer mittels Push ausliefert und das Datenpaket vom Datengebersystem entweder mittels "snapshotPull" oder "snapshotPush" eingeliefert wurde.
    b. "deltaPush", wenn die Mobilithek das Datenpaket an den Datennehmer mittels Push ausliefert und das Datenpaket vom Datengebersystem entweder mittels "deltaPull" oder "deltaPush" eingeliefert wurde.
    c. „snapshotPull“, wenn der Datennehmer eine Pull Anfrage an die Mobilithek schickt und das Datenpaket vom Datengebersystem entweder mittels "snapshotPull" oder "snapshotPush" eingeliefert wurde.
    d. "deltaPull", wenn der Datennehmer eine Pull Anfrage an die Mobilithek schickt und das Datenpaket vom Datengebersystem entweder mittels "deltaPull" oder "deltaPush" eingeliefert wurde.
4. SOAP-Pull Anfragen liefern immer nur das letzte vom Datengeber eingelieferte, vollständige Datenpaket aus. Delta-Datenpakete können mittels SOAP-Pull nicht von der Mobilithek bezogen werden.
5. Der Wert des Elements <messageGenerationTimestamp> wird bei der Auslieferung der Mobilithek an das Datennehmersystem nicht neu gesetzt. Das heißt, der Timestamp ist ein Ende-zu-Ende-Wert, der dem Original-Timestamp des eingelieferten Datenpakets entspricht.

4.6 Verwendung der Schnittstellen

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

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

Tabelle 7: Operationsmodi der Mobilithek

Für einen Datenaustausch mit der Mobilithek muss grundsätzlich für alle Protokolle eine Transportverschlüsselung[1] mit TLS 1.2 oder TLS 1.3 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.

Die Mobilithek 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 die Mobilithek 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.

[1] Unterstützt werden die folgenden Cipher-Suiten:
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

4.6.1 Datengeberseite

Gegenüber der Datengeberseite (dem Publisher) tritt die Mobilithek 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.

4.6.2 Datennehmerseite

Gegenüber der Datennehmerseite (dem Subscriber) tritt die Mobilithek 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.

4.7 Fehlerbehandlung

4.7.1 Client Pull vom Datengeber

Wenn der Datengeber als Kommunikationsweg Pull konfiguriert hat und das Datengebersystem auf einen Pull Request der Mobilithek mit einem Fehler reagiert, wird der Pull Request nicht unmittelbar wiederholt. Die Mobilithek wird den nächsten Pull Request gemäß der vom Datengeber spezifizierten Zeitdauer ausführen

4.7.2 Publisher Push

Wenn der Datennehmer als Kommunikationsweg Push konfiguriert hat und das konfigurierte Datennehmersystem nicht erreichbar ist, wird die Mobilithek in zunehmend länger werdenden Intervallen, versuchen das Datennehmersystem zu erreichen. Erst wenn das Datennehmersystem technisch erreichbar ist, wird eine erneute Datenzustellung versucht. Der Abstand zwischen den Intervallen wächst exponentiell. Die technische Umsetzung ist protokollspezifisch und wird in den entsprechenden Kapiteln erläutert.

Antwortet das Datennehmersystem auf einen Übertragungsversuch mit einem HTTP Response Status 500 oder 4xx wird der Übertragungsversuch unmittelbar wiederholt. Schlägt dieser Übertragungsversuch ebenfalls fehl, wird eine erneute Datenübertragung erst nach Eingang der nächsten vollständigen Datenlieferungen durch den Datengeber durch die Mobilithek initiiert.

4.8 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. Gemäß HTTP Standard ist die Auflösung des Zeitstempels auf Sekundenebene. Die Mobilithek rundet den Zeitstempel, der beim Eingang des Datenpakets gesetzt wird, auf die nächste volle Sekunde auf.

Enthält der HTTP Request das Header-Field "If-Modified-Since" nicht, wird das zuletzt eingelieferte Datenpaket von der Mobilithek ausgeliefert.

Im Zusammenhang mit Publikationen, für die Delta-Unterstützung aktiviert ist, können unter Nutzung der Verwendung dieses Headers auch Datennehmer über die PULL Schnittstellen von der Bandbreitenreduktion profitieren, die durch die Nutzung von Delta-Datenpaketen möglich wird. Hierzu wird der erste PULL Request mit einem Zeitstempel weit in der Vergangenheit gestartet. Diese Anfrage liefert das älteste Datenpaket aus dem Datenpuffer aus, dass per Definition eine vollständiges Datenpaket ist (siehe Kapitel 4.3). In den folgenden PULL Requests wird dann jeweils der Zeitstempel aus der Header Element "Last-Modified" verwendet.

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
```

4.8.1 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 Mobilithek in die Lage zu versetzen, dieses Feature anwenden zu können!

4.8.2 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!

4.8.3 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 Datenformat-unabhängige Schnittstellen

5.1 Löschen des Publikationsinhaltes

Über diese Schnittstelle kann ein Datengebersystem die Mobilithek auffordern alle Inhalte einer Publikation zu löschen. Die Publikation selbst bleibt dabei erhalten.

Request an die Mobilithek

Das Datengebersystem schickt einen HTTPS DELETE-Request an die Mobilithek. Der dazugehörige Paketpuffer wird mit der übergebenen Publikations-ID im Request identifiziert.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/<Publikations-ID>
```

Response an den Datengeber

Das Brokersystem der Mobilithek erzeugt nach Erhalt des Requests eine HTTPS Response. Konnte der Request erfolgreich verarbeitet werden, antwortet das Brokersystem der Mobilithek mit einem Response Code 200 und einem leeren Response Body.

Im Fehlerfall enthält der Response Body einen Fehlertext.

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

	Beschreibung
Request	Request DELETE /mobilithek/api/v1.0/publication/2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] ▪ 400: Fehlerhafter Request, bei z.B. Spezifikation einer nicht numerischen Publikations-ID ▪ 403: Das Datengebersystem hat keine Berechtigung den Inhalt der spezifizierten Publikation zu löschen ▪ 404: Publikation wurde nicht gefunden, oder es wurde keine Publikations-ID spezifiziert ▪ 503: Service Unavailable (z. B. bei Wartung)

Tabelle 8: Request/Response zwischen Mobilithek/Datengebersystem beim Löschen des Publikationsinhaltes

6 DATEX II v2

6.1 SOAP-Schnittstelle

6.1.1 Datengeberseite

6.1.1.1 Client Pull SOAP

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

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 Brokersystem der Mobilithek die angeforderten Daten im DATEX II Format zurück: im body-Element wird ein Objekt vom Typ d2LogicalModel erwartet. Gemäß dem Hinweis in Kapitel 4 akzeptiert das Brokersystem der Mobilithek alle Datenpakete, sofern diese in Form eines validen XMLs bereitgestellt werden. Das Brokersystem übernimmt den Inhalt des SOAP Bodies als Datenpaket.

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

Aufrufen eines Webservices

Das Brokersystem der Mobilithek 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-Endpunkts 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.

6.1.1.2 Publisher Push SOAP

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

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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 (Acknowledge Nachricht, siehe unten).

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

Die URL ist folgendermaßen aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/soap/<publication ID>/supplierPushService
```

Für den Fall, dass eine numerische Publikations-ID angegeben wurde, diese aber nicht existiert oder nicht aktiv ist, antwortet die Mobilithek mit einem HTTP Response Code 200 und einer SOAP Response mit dem denyReason "wrongCatalogue" und dem Response Code "requestDenied":

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" >
    <soapenv:Body>
        <d2LogicalModel xmlns="http://datex2.eu/schema/2/2_0" modelBaseVersion="2">
            <exchange>
                <denyReason>wrongCatalogue</denyReason>
                <response>requestDenied</response>
                <supplierIdentification>
                    <country>de</country>
                    <nationalIdentifier>Mobilithek.info</nationalIdentifier>
                </supplierIdentification>
            </exchange>
        </d2LogicalModel>
    </soapenv:Body>
</soapenv:Envelope>
```

Enthält das DATEX II Element im <exchange>-Element" ein Element <keepAlive> mit dem Wert true, wird keine Payload erwartet. Die Mobilithek antwortet mit einer Acknowledge Nachricht.

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

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

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 publikationsspezifischen Service-Endpunkt des Brokersystems der Mobilithek die Daten anliefern. Die zu verwendende URL des Service-Endpunkts wird beim Anlegen der Publikation in der Publikations-Konfiguration der Administrations-Komponente der Mobilithek angezeigt. Das Brokersystem der Mobilithek nimmt diese Daten an und speichert sie in einem Paketpuffer. Ein ggf. noch vorhandenes vorhergehendes Datenpaket wird dabei ersetzt.

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
200	denyReason: requestDenied response: unknownReason	Invalides XML oder invalider SOAP Request; Fehler beim Entpacken des Requests
		die Publikation ist keine DATEX2 V2 Publikation bzw. das vom Datengeber spezifizierte Protokoll ist nicht PUSH SOAP.

HTTP Response Code	SOAP Response	Beschreibung
	denyReason: requestDenied response: wrongCatalogue	Es existiert keine aktive Publikation mit der spezifizierten Publikations-ID
400	leerer Body	Die spezifizierte Publikations-ID ist nicht numerisch.
403	leerer Body	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Publikation bereitzustellen.
404	leerer Body	Keine Publikations-ID spezifiziert

Tabelle 9: Fehlercodes für DATEX II V2 - Publisher Push SOAP

6.1.2 Datennehmerseite

6.1.2.1 Client Pull SOAP

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

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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 wird ein Objekt vom Typ d2LogicalModel erwartet. Aufgrund der übermittelten Subskriptions-ID kann die Mobilithek den zugehörigen Paketpuffer sowie das Datenpaket ermitteln.

Beispiel:

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

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-Endpunkt des Brokersystems lautet:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/soap/<Subskriptions-ID>/clientPullService
```

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
200	faultcode: Server faultstring: Datex-II ClientPull - no data	Der zugehörige Paketpuffer enthält kein Datenpaket
400	leerer Body	Die spezifizierte Subskriptions-ID ist nicht numerisch
		Erforderliche HTTP Header fehlen, z.B. "Accept-encoding" Header ist nicht spezifiziert.
		HTTPS Request Body ist leer
404	leerer Body	Keine Subskriptions-ID spezifiziert
406	leerer Body	Im accept-encoding Header fehlt gzip als akzeptiertes Encoding
500	faultcode: Server faultstring: Contract can not be found, is not active or not available for provided orgId	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten unter dieser Subskription zu beziehen oder die spezifizierte Subskription existiert nicht.
	faultcode:Server faultString: Offer validation not passed reason: Access protocol, data model don't match	Die der Subskription zugehörige Publikation ist keine DATEX2 V2 Publikation.

Tabelle 10: Fehlercodes für Datennehmer DATEX II V2 Pull SOAP

6.1.2.2 Publisher Push SOAP

Beim Publisher Push Austauschverfahren liefert das Brokersystem der Mobilithek 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 der Mobilithek angeliefert werden, ist dabei unerheblich, der Mechanismus zur Abgabe an den Datennehmer ist identisch.

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 Mobilithek Bestätigungsdaten im DATEX II Format zurück. Dabei wird jeweils im body-Element ein Objekt vom Typ d2LogicalModel erwartet.

Aufrufen des Webservices

Das Brokersystem stellt einen auf Basis von [DATEXIIv2Push] definierten Webservice-Client zum Aufruf der Datennehmer-Webservices bereit. Über die Administrations-Komponente der Mobilithek muss der Datennehmer seinen Service-Endpunkt 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 einer solchen sog. Acknowledge-Response, welche bei Übertragung mit dem SOAP-Protokoll im Body-Element enthalten ist.

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
        <d2LogicalModel xmlns="https://datex2.eu/schema/2/2_0" modelBaseVersion="2">
            <exchange>
                <response>acknowledge</response>
            </exchange>
            <supplierIdentification>
                <country>de</country>
                <nationalIdentifier>DE-NAP-(ORGANISATIONSNAME)</nationalIdentifier>
            </supplierIdentification>
        </d2LogicalModel>
  </soapenv:Body>
</soapenv:Envelope>
```

Quittiert das Datennehmersystem die Übertragung nicht mit einer Acknowledge-Response wird gemäß Kapitel 4.7.2 die Übertragung wiederholt.

Ist das Datennehmersystem technisch nicht erreichbar, sendet die Mobilithek "keep-alive" Nachrichten an das Datennehmersystem, bis dieses mit einer Acknowledge-Response antwortet. Erst dann wird die der Versand von Daten fortgesetzt. Das nachstehende Beispiel zeigt eine "keep-alive" Nachricht:

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

6.2 HTTPS-Schnittstelle

6.2.1 Datengeberseite

6.2.1.1 Client Pull HTTPS

Beim Client Pull Austauschverfahren fordert das Brokersystem der Mobilithek zyklisch das Datengebersystem auf, seine Daten an der Mobilithek 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 die Mobilithek 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 bei der Mobilithek nicht zur Anwendung kommt. Rückgabewerte sind abweichend zu C.15 geregelt. Die Regel zum Aufbau der URLs auf die Payload (C.16) findet keine Anwendung. Siehe hierzu auch Anhang B – DATEX II HTTP Protokollunterstützung.

Request an den Datengeber

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

Über die Administrations-Komponente der Mobilithek muss der Datengeber die publikationsspezifische Server-URL in der Publikations-Konfiguration hinterlegen. Die URL muss vom Datengeber vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Publikations-ID.

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since“ Header-Feldes“.

Response an die Mobilithek

Das Datengebersystem muss nach Erhalt des Requests eine HTTPS Response erzeugen, deren Message-Body aus den angeforderten DATEX II Daten besteht. Gemäß [DATEXIIv2PSM] Kapitel 4 hat die Response den Content–Type „text/xml; charset=utf-8“ und kann als GZIP-Encoding vorliegen.

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

6.2.1.2 Publisher Push HTTPS

Das Datengebersystem muss ein Datenpaket zu einer Publikation an das Brokersystem der Mobilithek schicken.

Request an das Brokersystem der Mobilithek

Das Datengebersystem muss einen HTTPS POST-Request mit einer Nachricht im HTTP Request Body an das Brokersystem der Mobilithek schicken.

Dabei muss die Publikations-ID als Pfad-Element in der URL spezifiziert werden. Die Nutzdaten werden im HTTP Request Body übergeben

Der zu sendende Content-Type Header richtet sich nach der Syntax, auf die das entsprechende Datenangebot eingestellt ist.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/datexv2/<publicationID>
```

Es ist zu beachten, dass die HTTPS-Schnittstelle keine „keep-alive“ Nachrichten unterstützt.

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 Bedeutungen in Tabelle 25 gelten.

Beschreibung	
Request	Request POST /mobilithek/api/v1.0/publication/datexv2/<publicationID> HTTP/1.1 Host: mobilithek.info Content-Type: text/xml oder application/xml oder application/json Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: ▪ 400: Der angegebene Publikationsparameter ist nicht numerisch bzw. der Request ist nicht korrekt aufgebaut ▪ 404: Publikationsparameter konnte nicht zugeordnet werden, die Publikation ist nicht mehr gültig oder im URL Pfad wurde hinter dem Slash kein Wert angegeben ▪ 403: Der Benutzer ist nicht autorisiert für die angegebene Publikation Daten über diesen Endpunkt einzuliefern oder die Publikation wurde nicht zur Bereitstellung über HTTPS konfiguriert.

Tabelle 11: Request/Response zwischen Datengebersystem/Mobilithek beim Publisher Push DatexIIv2 HTTPS

6.2.2 Datennehmerseite

6.2.2.1 Client Pull HTTPS

Beim Client Pull Austauschverfahren muss das Datennehmersystem das Brokersystem der Mobilithek auffordern, die Daten zu übermitteln.

Request an die Mobilithek

Das Datennehmersystem muss einen HTTPS GET-Request an die URL der Mobilithek 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://mobilithek.info:8443/mobilithek/api/v1.0/subscription/<Subskriptions-ID>/clientPullService?subscriptionID=<Subskriptions-ID>
```

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since“ Header-Feldes“.

Response an den Datennehmer

Das Brokersystem der Mobilithek 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] Kapitel 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 12 beschriebenen Bedeutungen gelten.

	Beschreibung
Request	Request GET /mobilithek/api/v1.0/subscription/2000000/clientPullService?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info 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: ▪ 204: Kein Datenpaket im Paketpuffer zur Subskription. ▪ 304: Kein Datenpaket im Paketpuffer das jünger als der Zeitstempel im "if-modified-since" header ist. ▪ 400: Kein Subskriptionsparameter im Request spezifiziert oder fehlender "Accept-Encoding" Header oder Subskriptionsparameter ist nicht numerisch ▪ 403: Der Benutzer ist nicht autorisiert diese Subskription über diesen Endpunkt abzurufen oder es handelt sich nicht um eine DATEX2 V2 Publikation, die mit dieser Subskription verbunden ist. ▪ 404: Subskription ist nicht mehr gültig oder nicht existent. ▪ 406: GZIP nicht im "Accept-Encoding" Request Header spezifiziert.

Tabelle 12: Request/Response zwischen Mobilithek/Datennehmersystem bei Client Pull HTTPS

6.3 OCIT-C-Schnittstelle

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

6.3.1 Funktionsumfang

Die Mobilithek 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. Der Austausch von Teilmengen von Daten (Delta-Lieferungen) wird nicht unterstützt. Historische Daten können ebenfalls nicht abgefragt werden.

Die Mobilithek implementiert einen Webservice mit der vollständigen WSDL OCIT_Cif.wsdl, der unter spezifischen OCIT Endpunkten erreichbar ist. Der Aufruf einer nicht unterstützten Operation wird allerdings mit einem unmodelled SOAP-Fault (HTTP response code 500) mit dem Wert „Method not found“ 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 der Mobilithek 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

6.3.2 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 der Mobilithek 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 von der Mobilithek mit einem Fehler zurückgewiesen. Die Anlieferung eines DATEX II Elements muss immer vollständig sein, also alle Datenpunkte bzw. Objekte der Publikation enthalten. Die Mobilithek wird dies jedoch nicht überprüfen. Es liegt in der Verantwortung des Datengebersystems, die Vollständigkeit sicher zu stellen.

In der Metadatenverwaltung der Mobilithek kann das DATEX II Element manuell gegen das in der Mobilithek 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="Mobilithek"
                                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/
                                 Mobilithek-Profile_ParkingFacilityStatus.xsd">
              <ns2:exchange>
                <ns2:supplierIdentification>
                  <ns2:country>de</ns2:country>
                  <ns2:nationalIdentifier>DE-Mobilithek-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 die Mobilithek die folgenden Elemente aus dem OCIT-C-Protokoll:

- username
- password
- identifier innerhalb des putds-Attributs

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

- lastStart = Zeitpunkt der Anlieferung in der Mobilithek
- 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>
```

Fehlercodes

Mit Ausnahme der Situation, dass ein invalider SOAP Request gestellt wird, antwortet die Mobilithek bei Fehlern mit einer HTTP Response 200 und einem entsprechenden Fehlercode und -text. Die folgende Tabelle beschreibt die Fehlercodes und -situationen.

HTTP Response Code	SOAP Response	Beschreibung
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalides XML oder invalider SOAP Request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Mehrer SOAP Methoden im Request identifiziert, oder das erste Element im SOAP Body ist keine der unterstützten SOAP Methoden: <put>, <inquireAll>, <get> oder <waitForGet>
200	errorCode: 1 errorText: access error – erroneous object type	Die spezifizierte Publikations-ID ist nicht numerisch
	errorCode: 1 errorText: access error - exactly one putds must be present	Der Request enthält mehr als ein <putds>-Element
	errorCode: 14 errorText: found empty object type	Keine Publikations-ID im <objectType>-Element gefunden
	errorCode: 15 errorText: object type not found – is missing	<objectType> Element im Request nicht gefunden.
	errorCode: 1 errorText: access error	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Publikation bereitzustellen, oder die spezifizierte Publikation ist keine DATEX2 V2 Publikation, oder OCIT-C wurde nicht als Zugangsprotokoll definiert.
	errorCode: 1 errorText: access error - no valid certificate-publication match	Die spezifizierte Publikations-ID existiert nicht.
	errorCode: 1 errorText: access error - exactly one putds must be present	Der SOAP Request unterstützt genau ein <putds> Element

Tabelle 13: Fehlercodes für OCIT-C Publisher Push

6.3.3 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. Die Mobilithek unterstützt zu diesem Zweck die inquireAll-Methode. In der inquireAllResponse übergibt die Mobilithek das letzte gültige Paket und eine 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 in der Mobilithek kein neues Paket vor, kehrt die get-Methode sofort mit einer leeren Antwort zurück. Die wait4Get-Methode wartet so lange, 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 die Mobilithek mit einer get- bzw. wait4GetResponse immer ein vollständiges Datenpaket zurück und nicht nur Delta-Daten bezogen auf die letzte Position. Die Mobilithek unterstützt für DATEX II v2 keine Delta-Pakete.

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 die Mobilithek 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 der Mobilithek 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 die Mobilithek 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 in der Mobilithek
- position = Paket-ID, ID des aktuellen Datenpakets, nur von Bedeutung für die hier spezifizierten OCIT-C Methoden
- 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="Mobilithek"
                                     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/
                                Mobilithek-Profile_ParkingFacilityStatus.xsd">
                        <exchange>
                           <supplierIdentification>
                              <country>de</country>
                                 <nationalIdentifier>DE-Mobilithek-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 Paket-ID zugeordnet werden. Diese Subskriptions-ID muss das Datennehmersystem im OCIT-C-Attribut <objectType> übergeben, die Paket-ID im Attribut <position>. Ein get-Aufruf unter Nutzung von Start- und Endezeit unterstützt die Mobilithek 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-Paket-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>
```

Die Mobilithek 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 in der Mobilithek konfigurierten Maximalwert von 120 Sekunden, so wird der Mobilithek-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 von der Mobilithek 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 der Mobilithek:

```
<?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 an der Mobilithek 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.

Fehlercodes

Mit Ausnahme der Situation, dass ein invalider SOAP Request gestellt wird, antwortet die Mobilithek bei Fehlern mit einer HTTP Response 200 und einem entsprechenden Fehlercode und -text. Die folgende Tabelle beschreibt die Fehlercodes und -situationen.

HTTP Response Code	SOAP Response	Beschreibung
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalides XML oder invalider SOAP Request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Mehrer SOAP Methoden im Request identifiziert, oder das erste Element im SOAP Body ist keine der unterstützten SOAP Methoden: <put>, <inquireAll>, <get> oder <waitForGet>
	errorCode: 0 keine Daten	Keine Datenpakete verfügbar, oder keine Datenpakete gemäß des im Request spezifizierten <position>-Elements verfügbar
400	leerer Body	fehlender "Accept-Encoding" Header oder Subskriptionsparameter ist nicht numerisch
406	leerer Body	Im accept-encoding Header fehlt gzip als akzeptiertes Encoding
200	errorCode: 1 errorText: access error – erroneous object type	Die spezifizierte Subskriptions-ID ist nicht numerisch
	errorCode: 14 errorText: found empty object type	Keine Subskriptions-ID im <objectType> Element gefunden
	errorCode: 15 errorText: object type not found – is missing	<objectType> Element im Request nicht gefunden.
	errorCode: 1 errorText: access error – no valid certificate-subscription match	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Subskription zu beziehen, oder die spezifizierte Subskription ist keine DATEX2 V2 Publikation
		Die spezifizierte Subskriptions-ID existiert nicht.

HTTP Response Code	SOAP Response	Beschreibung
	errorCode: 1 errorText: access error - exactly one position must be present	Der Aufruf des <get> oder <wait4Get> Requests erfordert verpflichtend die Angabe <position> Elements mit einem numerischen Wert. Dieser wurde nicht gemäß WSDL spezifiziert.

Tabelle 14: Fehlercodes für OCIT-C Datennehmer Pull

7 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 in der Mobilithek 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 (siehe auch Kapitel 4.5.2):
    - Für SOAP Schnittstellen wird hier entweder „snapshotPull“, „snapshotPush“, "deltaPull" oder "deltaPush" verwendet.
    - Für HTTP pull wird hier „snapshotPull“ oder "deltaPull" verwendet.
  - <exchangeSpecificationVersion>: Die Mobilithek 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.

Über das Element <codedExchangeprotocol> wird somit angezeigt, ob es sich bei einem DATEX II v3 Datenpaket um ein vollständiges oder ein Delta-Paket handelt. Die Behandlung von Delta-Paketen wird in Kapitel 4.3 beschrieben.

In der Administrations-Komponente der Mobilithek kann der Datengeber auswählen, ob für diese Publikation eine Anlieferung von Delta-Paketen vorgesehen ist.

> **Wichtiger Hinweis**
>
> Werden für eine DATEX II v3 Publikation Delta-Pakete angeliefert, können Datennehmer, diese nicht mittels Pull Request über SOAP abholen. SOAP Pull Requests werden immer das zuletzt angelieferte vollständige Datenpaket ausliefern.

7.1 Hinweise zur Behandlung von Schemas mit Exchange 2020

Exchange 2020 unterstützt die Bereitstellung von Publikationen in XML oder JSON Format.

Will man eine Publikation für DATEX II v3 Inhalte einrichten, muss der Datengeber mehrere Schemas (XML oder JSON Schema) 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 Schema erzeugt. Jede Instanz der Publikation muss das Einstiegsschema referenzieren, welches je nach Kompatibilitätsstufe entweder DATEXII_3_D2Payload.[xsd | json] (Level A oder B) oder LevelC_3_D2Payload.[xsd | json] (Level C) heißt. Dieses Schema importiert alle weiteren 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 der Mobilithek umgesetzten Optionen in drei weitere Schemas eingebettet: MessageContainer.[xsd | json], InformationManagement.[xsd | json] und ExchangeInformation.[xsd | json].

Die Protokolldatenschemas wurden für die Anwendung im Rahmen der Mobilithek 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

7.1.1 DATEX II v3 Level A oder B

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

7.1.2 DATEX II v3 Level C

Bei der Erzeugung der Schemas des Datenprofiles der Publikation werden mehrere Schemas erzeugt. Das in die Instanzen der Publikation einzubettende Schema heißt LevelC_3_D2Payload.[xsd | json] und definiert in XML den Namenraum http://levelC/schema/3/d2Payload. Der Datengeber muss diese Inhaltsdatenschemas zusammen mit der Variante von ExchangeInformation.[xsd | json] sowie

MessageContainer.[xsd | json] für Level C in der Benutzeroberfläche zu seiner Publikation hochladen [DATEXIIv3Exc].

Wichtige Hinweise

1. DATEX II v3 Inhalte auf der Mobilithek müssen immer auf einem von der abstrakten Klasse PayloadPublication im DATEX II v3 Paket Common abgeleiteten 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 Schema muss immer Bestandteil der Inhaltsdatenschemas sein, also entweder DATEXII_3_Common.[xsd | json] (Level A oder B) oder LevelC_3_Common.[xsd | json] (Level C).
   Benutzer, die eine Level C-Publikation mit anderer Struktur anstreben, müssen entsprechende manuelle Anpassungen auf Schema-Ebene durchführen. Bei Bedarf sollten sie sich für Unterstützung an den Mobilithek-Support wenden.
2. Aus technischen Gründen müssen referenzierte Subschemas in JSON mit dem file: Präfix begonnen werden. Zum Beispiel:

   "$ref": "file:LevelC_3_Common.json#/definitions/_ExtensionType"

7.2 SOAP-Schnittstelle

7.2.1 Datengeberseite

7.2.1.1 Client Pull SOAP

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

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 Brokersystem der Mobilithek 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“ für ein vollständiges Datenpaket
- "deltaPull" für ein Delta-Datenpaket

gesetzt werden soll.

Unabhängig davon ersetzt die Mobilithek das <codedExchangeProtocol>-Element mit dem jeweils dem Auslieferungsprotokoll entsprechenden Wert (siehe Kapitel 4.5.2), um standardkonformen Datennehmersystemen eine korrekte Verarbeitung zu ermöglichen. Dem Hinweis in Kapitel 4 folgend, nimmt das Brokersystem nur notwendige Validierungen der Response vor. Damit das Datenpaket von der Mobilithek übernommen wird, muss es ein valides XML Format aufweisen und das oben spezifizierte Element <codedExchangeProtocol> mit einem gültigen Wert belegt sein.

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

Aufrufen eines Webservices

Das Brokersystem der Mobilithek 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 Brokersystem der Mobilithek folgende Antwort:

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

Beispiel-Response:

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <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="<URL des Schema Files>"
                          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>
  </soapenv:Body>
</soapenv:Envelope>
```

Das Brokersystem identifiziert die Datengebersysteme, die ein Pull-Verfahren abonniert haben, sowie die zugehörigen Service-Endpunkts 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. Ist das angelieferte Datenpaket ein vollständiges Paket werden alle im Paketpuffer befindlichen Datenpakete ersetzt. Ein Delta-Paket wird an die Liste der im Paketpuffer befindlichen Datenpakete angehängt.

7.2.1.2 Publisher Push SOAP

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

geliefert werden, ist für die Funktionsweise des Brokersystems der Mobilithek unerheblich. Der Mechanismus zum Austausch ist in beiden Fällen identisch.

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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

- „snapshotPull“ für ein vollständiges Datenpaket
- "deltaPull" für ein Delta-Datenpaket

gesetzt werden soll.

Unabhängig davon ersetzt die Mobilithek das <codedExchangeProtocol>-Element mit dem jeweils dem Auslieferungsprotokoll entsprechendem Wert (siehe Kapitel 4.5.2), um standardkonformen Datennehmersystemen eine korrekte Verarbeitung zu ermöglichen.

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

Die URL ist folgendermaßen aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/soap/datexv3/<publication ID>/snapshotPushService
```

Beispiel:

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
      <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="<URL des Schema Files>"
                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>
    </soapenv:Body>
</soapenv:Envelope>
```

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-Endpunkt des Brokersystems der Mobilithek die Daten anliefern. Das Brokersystem nimmt diese Daten an und speichert sie in einem Paketpuffer. Ist das angelieferte Datenpaket ein vollständiges Paket werden alle im Paketpuffer befindlichen Datenpakete ersetzt. Ein Delta-Paket wird an die Liste der im Paketpuffer befindlichen Datenpakete angehängt.

Eine erfolgreiche Anlieferung beantwortet die Mobilithek 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 <URL des Schema
Files>" 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 die Mobilithek 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. In der Response werden folgende Werte für das Element <codedInvalidityReason> verwendet:

- invalidPayload, falls keine Publikation zur angegebenen ID gefunden wurde.
- invalidMessage, falls der SOAP Request nicht korrekt entpackt werden konnte.

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 <URL des Schema
Files>" 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:codedInvalidityReason>invalidPayload</ex:codedInvalidityReason>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
200	returnStatus: fail codedInvalidityReason:invalidMessage	Invalides XML oder invalider SOAP Request; Fehler beim Entpacken des Requests, oder Die Publikation ist keine DATEX II V3 Publikation bzw. das vom Datengeber spezifizierte Protokoll ist nicht PUSH SOAP.
	returnStatus: fail codedInvalidityReason: invalidPayload	Das Element <codedExchangeProtocol> wurde nicht gemäß WSDL spezifiziert bzw. enthält nicht den Wert "snapshotPush" oder "deltaPush", oder die Publikations-ID existiert nicht
400	leerer Body	Die spezifizierte Publikations-ID ist nicht numerisch.
403	leerer Body	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Publikation bereitzustellen.
404	leerer Body	Keine Publikations-ID spezifiziert

Tabelle 15: Fehlercodes für DATEX2 V3 Publisher Push SOAP

7.2.2 Datennehmerseite

7.2.2.1 Client Pull SOAP

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

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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 Mobilithek den zugehörigen Paketpuffer sowie das Datenpaket ermitteln.

> **Wichtiger Hinweis**
>
> - Enthält der Paketpuffer zum Zeitpunkt der Anfrage kein Datenpaket beantwortet die Mobilithek die Anfrage mit einem MessageContainer ohne payload-Element. (vgl. Aufrufen eines Webservices)
> - Auch wenn für die Publikation durch den Datengeber Delta-Pakete eingeliefert werden, liefert der SOAP Request immer nur das zuletzt eingelieferte, vollständige Datenpaket zurück.

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-Endpunkt des Brokersystems lautet:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/soap/datexv3?subscriptionId=<Subskriptions-ID>
```

Beispiel:

```
<?xml version='1.0'?>
<soapenv:Envelope xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'
                  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
                  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
  <soapenv:Body>
    <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 <URL des
Schema Files>'
                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>
  </soapenv:Body>
</soapenv:Envelope>
```

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
200	DATEX2 V3 Antwort ohne <payload>-Element	Der zugehörige Paketpuffer enthält kein Datenpaket
405	leerer Body	Der Parameter ?subscriptionId= wurde nicht spezifiziert bzw. es wurde kein Wert für den Parameter spezifiziert
406	leerer Body	Im accept-encoding Header fehlt gzip als akzeptiertes Encoding
500	faultcode: Client faultstring: Contract can not be found, is not active or not available for provided orgId	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten unter dieser Subskription zu beziehen oder die spezifizierte Subskription existiert nicht.
	faultcode:Client faultstring: invalid - XML	Invalides XML oder invalider SOAP Request
	faultcode:Client faultString: Offer validation not passed reason: Access protocol, data model don't match	Die der Subskription zugehörige Publikation ist keine DATEX2 V3 Publikation.

Tabelle 16: Fehlercodes für Datennehmer DATEX II V3 Pull SOAP

7.2.2.2 Publisher Push SOAP

Beim Publisher Push Austauschverfahren liefert das Brokersystem der Mobilithek 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 bei der Mobilithek angeliefert werden, ist dabei unerheblich, der Mechanismus zur Abgabe an den Datennehmer ist identisch.

Anbieten eines Webservices

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

Aufrufen des Webservices

Die Mobilithek stellt einen auf Basis von [DATEXIIv3Push] definierten Webservice-Client zum Aufruf der Datennehmer-Webservices bereit. Über die Mobilithek-Administrations-Komponente muss der Datennehmer seinen Service-Endpunkt 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:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <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="<URL des Schema Files>"
            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>
  </soapenv:Body>
</soapenv:Envelope>
```

Synchronisation

Ein Datennehmersystem kann die Mobilithek veranlassen Datenpakete erneut zu verschicken. Dies kann insbesondere nützlich sein, um im Zusammenhang mit Publikationen, die Delta-Lieferungen unterstützen, nach einem Start des Datennehmersystems eine Synchronisation auf den aktuellen Stand des Datenpaketpuffers der Mobilithek durchzuführen und nicht auf die nächste Lieferung eines vollständigen Datenpakets warten zu müssen.

Erhält die Mobilithek als Response auf eine Datenübertragung eine Bestätigungsnachricht mit dem Return Status "**snapshotSynchronisationRequest**", wird die Mobilithek alle im Datenpaketpuffer enthaltenen Datenpakete in der Reihenfolge des Empfangs erneut an das Datennehmersystem übertragen. Im Folgenden ist eine Bestätigungsnachricht, die eine Synchronisation einleitet, dargestellt.

```
<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="<URL des Schema Files>"
        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>snapshotSynchronisationRequest</ex:returnStatus>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

7.3 HTTPS-Schnittstelle

7.3.1 Datengeberseite

7.3.1.1 Client Pull HTTPS

Beim Client Pull Austauschverfahren fordert das Brokersystem der Mobilithek zyklisch das Datengebersystem auf, seine Daten an der Mobilithek 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*.

Dabei 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 die Mobilithek verpflichtenden HTTPS-Verfahrens obsolet sind. Siehe hierzu auch Anhang B – DATEX II HTTP Protokollunterstützung.

Request an den Datengeber

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

Über die Administrations-Komponente der Mobilithek muss der Datengeber die publikationsspezifische Server-URL in der Publikations-Konfiguration hinterlegen. Die URL muss vom Datengeber vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Publikations-ID.

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since" Header-Feldes".

Response an die Mobilithek

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 Brokersystem der Mobilithek nimmt diese Daten an und speichert sie in einem Paketpuffer. Ist das angelieferte Datenpaket ein vollständiges Paket werden alle im Paketpuffer befindlichen Datenpakete ersetzt. Ein Delta-Paket wird an die Liste der im Paketpuffer befindlichen Datenpakete angehängt.

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 <URL des Schema
Files>'
            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>
```

7.3.1.2 Publisher Push HTTPS

Das Datengebersystem muss ein Datenpaket zu einer Publikation an das Brokersystem der Mobilithek schicken.

Request an das Brokersystem der Mobilithek

Das Datengebersystem muss einen HTTPS POST-Request mit einer Nachricht im HTTP Request Body an das Brokersystem der Mobilithek schicken.

Dabei muss die Publikations-ID als Pfad-Element in der URL spezifiziert werden. Die Nutzdaten werden im HTTP Request Body übergeben

Der zu sendende Content-Type Header richtet sich nach der Syntax, auf die das entsprechende Datenangebot eingestellt ist.

Es liegt in der Verantwortung des Datengebers, das verpflichtende <exchangeInformation>-Element mit seinen Elementen <exchangeContext> und <dynamicInformation> zu definieren, sofern es sich um eine Delta-Anlieferung handelt. Um standardkonform die Daten bereitzustellen, ist zu beachten, dass das Element <codedExchangeProtocol> auf den Wert

- „snapshotPull“ für ein vollständiges Datenpaket
- "deltaPull" für ein Delta-Datenpaket

gesetzt werden soll. Dafür ist es notwendig, dass das Datenpaket ein valides XML Format (oder JSON Format, wenn das Datenangebot mit der Syntax JSON eingestellt worden ist) aufweist. Wenn das Datenangebot keine Deltalieferung vorsieht, wird nicht nach der Existenz oder dem Inhalt des Elements <codedExchangeProtocol> geprüft.

Unabhängig davon ersetzt die Mobilithek das <codedExchangeProtocol>-Element mit dem jeweils dem Auslieferungsprotokoll entsprechenden Wert (siehe Kapitel 4.5.2), um standardkonformen Datennehmersystemen eine korrekte Verarbeitung zu ermöglichen. Dem Hinweis in Kapitel 4 folgend, nimmt das Brokersystem nur notwendige Validierungen der Response vor.

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/datexv3/<publicationID>/snapShotPushService
```

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 Bedeutungen in Tabelle 25 gelten.

Beschreibung	
Request	Request POST /mobilithek/api/v1.0/publication/datexv3/<publicationID>/snapShotPushService HTTP/1.1 Host: mobilithek.info Content-Type: text/xml oder application/xml oder application/json Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: ▪ 400: Der angegebene Publikationsparameter ist nicht numerisch bzw. der Request ist nicht korrekt aufgebaut ▪ 404: Publikationsparameter konnte nicht zugeordnet werden, die Publikation ist nicht mehr gültig oder im URL Pfad wurde hinter dem Slash kein Wert angegeben ▪ 403: Der Benutzer ist nicht autorisiert für die angegebene Publikation Daten über diesen Endpunkt einzuliefern oder die Publikation wurde nicht zur Bereitstellung über HTTPS konfiguriert. ▪ 422: Für ein Angebot, welches Deltaanlieferungen unterstützt, wurde kein gültiger Wert für das Element codedExchangeProtocol übergeben oder die Syntax der übergebenen Nachricht in invalide

Tabelle 17: Request/Response zwischen Datengebersystem/Mobilithek beim Publisher Push DatexIIv3 HTTPS

7.3.2 Datennehmerseite

7.3.2.1 Client Pull HTTPS

Beim Client Pull Austauschverfahren muss das Datennehmersystem die Mobilithek auffordern, die Daten zu übermitteln.

Request an die Mobilithek

Das Datennehmersystem soll einen HTTPS GET-Request an die Mobilithek 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 für XML basierte Publikationen ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/datexv3?subscriptionID=<Subskriptions-ID>
```

Die URL des Brokersystems für JSON basierte Publikationen ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription?subscriptionID=<Subscriptions-Id>
```

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since“ Header-Feldes“.

Response an den Datennehmer

Das Brokersystem der Mobilithek 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=utf8“ bzw. "application/json; charset=utf8" 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 18 beschriebenen Bedeutungen gelten:

Beschreibung	
Request	Request GET /mobilithek/api/v1.0/subscription/datexv3?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info 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: ▪ 204: Kein Datenpaket im Paketpuffer zur Subskription ▪ 304: Kein Datenpaket im Paketpuffer das jünger als der Zeitstempel im "if-modified-since" header ist. ▪ 400: Kein Subskriptionsparameter im Request spezifiziert oder fehlender Accept-Encoding Header ▪ 403: Der Benutzer ist nicht autorisiert diese Subskription über diesen Endpunkt abzurufen oder es handelt sich nicht um eine DATEX2 V3 Publikation, die mit dieser Subskription verbunden ist. ▪ 404: Subskriptionsparameter ist nicht numerisch oder Subskription ist nicht mehr gültig oder nicht existent. ▪ 406: GZIP nicht im "Accept-Encoding" Request Header spezifiziert. ▪ 503: Service Unavailable (z. B. bei Wartung)

Tabelle 18: Request/Response zwischen Mobilithek/Datennehmersystem beim Client Pull HTTPS

8 Container

8.1 SOAP-Schnittstelle

8.1.1 Datengeberseite

8.1.1.1 Client Pull SOAP

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

Anbieten eines Webservices

Das Datengebersystem muss einen Webservice mit der Methode pullContainerDataBroker 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.

Über die Administrations-Komponente der Mobilithek muss der Datengeber den Service-Endpunkt im URL-Attribut der Publikations-Konfiguration hinterlegen. Das Brokersystem der Mobilithek geht dabei davon aus, dass die Datenpakete, die das Datengebersystem für die angegebene URL abgibt zu der konfigurierten Publikation gehören, unabhängig von der Publikations-ID im SOAP Request.

Das Datengebersystem muss ein Datenpaket (Typ containerdata) im Containerformat erzeugen und zurückschicken. Dem Hinweis in Kapitel 4 folgend, übernimmt das Brokersystem der Mobilithek alle Datenpakete, sofern diese ein valide XML Struktur aufweisen.

> **Wichtiger Hinweis**
>
> Die Administrations-Komponente der Mobilithek ermöglicht, die Erreichbarkeit der spezifizierten URL des Datengebers zu testen. Da dies zu einem Zeitpunkt möglich ist, zu der die ID der Publikation in der Mobilithek noch nicht bekannt ist, wird die Mobilithek diesen Test Request mit einer zufälligen ID ausführen. Das Datengebersystem muss daher unabhängig von der im Request enthaltenen Publikations-ID ein gültiges Datenpaket erzeugen.

Aufrufen eines Webservices

Das Brokersystem der Mobilithek 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 den zugehörigen Service-Endpunkt 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.

8.1.1.2 Publisher Push SOAP (Container)

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

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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'?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv: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>
  </soapenv:Body>
</soapenv:Envelope>
```

Aufrufen des Webservices

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

Der SOAP-Endpunkt des Brokersystems lautet:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/soap/container
```

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalides XML oder invalider SOAP Request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Mehrere SOAP Methoden im Request identifiziert, oder das erste Element im SOAP Body ist nicht die unterstützte SOAP Methode: <container>
	SOAP Fault faultcode: Client faultstring: numeric publicationID expected, found non-numeric publicationId	Die spezifizierte Publikations-ID ist nicht numerisch
	SOAP Fault faultcode: Client faultstring: tag <publicationID> expected according to WDSL, element not found	<publicationID> Element im Request nicht gefunden.
	SOAP Fault faultCode: Client faultString = publicationId for this organisation not found	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Publikation bereitzustellen, oder die spezifizierte Publikation existiert nicht, oder für die spezifizierte Publikation wurde nicht das Anlieferungsprotokoll HTTPS-SOAP ausgewählt

Tabelle 19: Fehlercodes für SOAP Container Publisher Push

8.1.2 Datennehmerseite

8.1.2.1 Client Pull SOAP

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

Anbieten eines Webservices

Das Brokersystem der Mobilithek 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'?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv: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>
  </soapenv:Body>
</soapenv:Envelope>
```

Aufrufen des Webservices

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

Der SOAP-Endpunkt des Brokersystems lautet:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/subscription/soap/container
```

Fehlercodes

HTTP Response Code	SOAP Response	Beschreibung
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalides XML oder invalider SOAP Request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Mehrer SOAP Methoden im Request identifiziert, oder das erste Element im SOAP Body ist nicht die unterstützte SOAP Methode: <pullContainerDataClientRequestEl>
	SOAP Fault faultcode: Client faultstring: Expected to find numerical value in subscriptionId, found non-numeric	Die spezifizierte Subskriptions-ID ist nicht numerisch
	SOAP Fault faultcode: Client faultstring: Expected to find element subscriptionId in request, but did not find	<subscriptionID>-Element im Request nicht gefunden.
	SOAP Fault faultCode: Client faultString: Accessible subscriptionId for organisation not found	Der dem Maschinen Account zugeordnete Benutzer hat keine Berechtigung Daten für die spezifizierte Subskription zu beziehen, oder die spezifizierte Subskription existiert nicht, oder die spezifizierte Subskription ist keine Container Publikation
	SOAP Fault faultCode: Server faultString: No data package available	Kein Datenpaket im Paketpuffer vorhanden
400	leerer Body	Request fehlerhaft, z.B. "Accept-Encoding" Header fehlt
406	leerer Body	gzip nicht im "Accept-Encoding" Request Header spezifiziert.

Tabelle 20: Fehlercodes für SOAP Container Consumer Pull

8.1.2.2 Publisher Push SOAP

Beim Publisher Push Austauschverfahren liefert die Mobilithek 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 bei der Mobilithek angeliefert werden, ist dabei unerheblich, der Mechanismus zur Abgabe an den Datennehmer ist identisch.

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.

Aufrufen des Webservices

Das Brokersystem der Mobilithek stellt einen auf Basis der Containerformat Spezifikation [MCS] definierten Webservice-Client zum Aufruf der Datennehmer-Webservices bereit. Über die Administrations-Komponente der Mobilithek muss der Datennehmer seinen Service-Endpunkt 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.

Quittiert das Datennehmersystem die Übertragung nicht als erfolgreich wird gemäß Kapitel 4.7.2 die Übertragung wiederholt.

Ist das Datennehmersystem technisch nicht erreichbar, werden HTTP Requests an den um das Suffix "?wsdl" erweiterten Service-Endpunkt gesendet. Erst wenn dieser HTTP Request mit einem Status 200 quittiert wird, wird das Datennehmersystem erneut beliefert.

8.2 HTTPS-Schnittstelle

8.2.1 Datengeberseite

8.2.1.1 Client Pull HTTPS

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

Request an den Datengeber

Das Brokersystem schickt einen HTTPS GET-Request zum Datengebersystem. Als URL benutzt das Brokersystem die in der Publikationsbeschreibung gespeicherte URL.

Beispiel:

```
GET <URL gemäß Publikationsbeschreibung>
content-type: text/plain
accept-encoding: gzip
```

Die URL muss vom Datengeber vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Publikations-ID

Response an die Mobilithek

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 Mobilithek verarbeitet werden. Der Message-Body muss aus dem angeforderten Datenpaket bestehen.

Die im Response Body übermittelte Payload wird nur im Brokersystem der Mobilithek gespeichert, wenn das Datengebersystem mit einem Status 200 antwortet.

Beachten Sie auch die Hinweise in Kapitel 4.8.1 „“ zur Nutzung der If-Modified-Since und Last-Modified HTTP Header.

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">
        </xml>
    </body>
</container>
```

8.2.1.2 Publisher Push HTTPS

Das Datengebersystem muss ein Datenpaket zu einer Publikation an das Brokersystem der Mobilithek schicken.

Request an das Brokersystem der Mobilithek

Das Datengebersystem muss einen HTTPS POST-Request mit einer Nachricht im Containerformat zum Brokersystem der Mobilithek 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://mobilithek.info:8443/mobilithek/api/v1.0/publication/container
```

Beispiel:

```
<?xml version='1.0' encoding='UTF-8'?>
<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:xml schema="test-schema" id="test-id-xml">
    </ns3:xml>
  </ns3:body>
</ns3:container>
```

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 21 beschriebenen Bedeutungen gelten.

Beschreibung	
Request	Request POST /mobilithek/api/v1.0/publication/container HTTP/1.1 Host: mobilithek.info Content-Type : application/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, der Publikationsparameter ist nicht numerisch oder der Request ist nicht korrekt aufgebaut, z.B. er enthält kein valides XML ▪ 403: Der Benutzer ist nicht autorisiert für die angegebene Publikation Daten über diesen Endpunkt einzuliefern oder die Publikation wurde nicht für die Einlieferung HTTPS Container spezifiziert ▪ 404: Publikationsparameter konnte keiner gültigen Publikation zugeordnet werden

Tabelle 21: Request/Response zwischen Datengebersystem/Mobilithek beim Publisher Push HTTPS

8.2.2 Datennehmerseite

8.2.2.1 Client Pull HTTPS

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

Request an die Mobilithek

Das Datennehmersystem muss einen HTTPS GET-Request an die Mobilithek 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://mobilithek.info:8443/mobilithek/api/v1.0/container/subscription?subscriptionID=<Subskriptions-ID>
```

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since“ Header-Feldes“.

Response an das Datennehmersystem

Das Brokersystem der Mobilithek erzeugt nach Erhalt des Requests eine HTTPS Response. Als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in Tabelle 22 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 /mobilithek/api/v1.0/container/subscription?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info 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. ▪ 304: Kein Datenpaket im Paketpuffer das jünger als der Zeitstempel im "if-modified-since" header ist. ▪ 400: Kein Subskriptionsparameter oder fehlender Accept-Encoding Header ▪ 403: Der Benutzer ist nicht autorisiert diese Subskription über diesen Endpunkt abzurufen oder die zugehörige Publikation wird nicht im Container Format bereitgestellt. ▪ 404: Keine oder eine nicht mehr gültige Subskription zum Subskriptionsparameter gefunden. ▪ 405: Der Parameter ?subscriptionId= wurde nicht spezifiziert bzw. es wurde kein Wert für den Parameter spezifiziert ▪ 406: gzip nicht im "Accept-Encoding" Request Header spezifiziert.

Tabelle 22: Request/Response zwischen Mobilithek/Datennehmersystem beim Client Pull HTTPS

8.2.2.2 Publisher Push HTTPS

Das Brokersystem der Mobilithek schickt ein Datenpaket zu einer Subskription an ein Datennehmersystem.

Request an das Datennehmersystem

Das Brokersystem der Mobilithek 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 Administrations-Komponente der Mobilithek muss der Datennehmer seine URL in der Subskriptions-Konfiguration hinterlegen.

Die URL muss vom Datennehmer vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Subskriptions-ID.

Response an die Mobilithek

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 23 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]

Tabelle 23: Request/Response zwischen Brokersystem der Mobilithek/Datennehmersystem beim Publisher Push HTTPS

Bezugnehmend auf die Ausführungen in Kapitel 4.7.2 führen andere Statuscodes als 200, zu einem wiederholten Übertragungsversuch des Datenpakets.

Ist das Datennehmersystem technisch nicht erreichbar, werden HTTP-Head Request an die hinterlegte URL geschickt. Erst wenn der HTTP-Head Request durch das Datennehmersystem mit einem Status 200 quittiert wird, wird es erneut beliefert.

9 Andere Datenformate

Die Mobilithek unterstützt die Anlieferung und Auslieferung beliebiger Datenformate, ohne die Notwendigkeit diese in das Containerformat einzupacken.

9.1 Datengeberseite

9.1.1 Client Pull HTTPS

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

9.1.1.1 Request an den Datengeber

Das Brokersystem schickt einen HTTPS GET-Request zum Datengebersystem. Als URL benutzt das Brokersystem die in der Publikationsbeschreibung gespeicherte URL.

Beispiel:

```
GET <URL gemäß Publikationsbeschreibung>
content-type: text/plain
accept-encoding: gzip
```

Die URL muss vom Datengeber vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Publikations-ID

9.1.1.2 Response an die Mobilithek

Auf den Request muss das Datengebersystem mit einer HTTPS-Response antworten. Der Content-Type der Response sollte dem Typ der übermittelten Payload entsprechen und sollte als GZIP-Encoding vorliegen. Auch nicht komprimierter Inhalt kann von der Mobilithek verarbeitet werden. Die Mobilithek führt keinerlei Prüfung hinsichtlich der Konsistenz des spezifizierten Content-Type und der übermittelten Payload durch.

Der Message-Body muss aus dem angeforderten Datenpaket bestehen. Als Statuscodes sind die Standard HTTP Statuscodes [HTTP/1.1] zu verwenden. Die im Response Body übermittelte Payload wird nur im Brokersystem der Mobilithek gespeichert, wenn das Datengebersystem mit einem Status 200 antwortet.

Beachten Sie auch die Hinweise in Kapitel 4.8.1 zur Nutzung der If-Modified-Since und Last-Modified HTTP Header.

Beschreibung	
Request	GET <URL gemäß Publikationsbeschreibung> HTTP/1.1 Host: Datengeberhost Accept-Encoding: gzip
Response	HTTP/1.1 200 OK Content-Type: text/csv Content-Length: xx [Datenpaket]
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1]

Tabelle 24: Request/Response zwischen Datengebersystem/Mobilithek beim Client Pull HTTPS

9.1.2 Publisher Push HTTPS

Das Datengebersystem muss ein Datenpaket zu einer Publikation an das Brokersystem der Mobilithek schicken.

9.1.2.1 Request an das Brokersystem der Mobilithek

Das Datengebersystem muss einen HTTPS POST-Request mit einer Nachricht im HTTP Request Body an das Brokersystem der Mobilithek schicken.

Dabei muss die Publikations-ID als Pfad-Element in der URL spezifiziert werden. Die Nutzdaten werden im HTTP Request Body übergeben

Die URL des Brokersystems ist wie folgt aufgebaut:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/publication/<publicationID>
```

9.1.2.2 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 Bedeutungen in Tabelle 25 gelten.

Beschreibung	
Request	Request POST /mobilithek/api/v1.0/publication/<publicationID> HTTP/1.1 Host: mobilithek.info Content-Type: text/csv Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: ▪ 400: Der angegebene Publikationsparameter ist nicht numerisch bzw. der Request ist nicht korrekt aufgebaut ▪ 404: Publikationsparameter konnte nicht zugeordnet werden, die Publikation ist nicht mehr gültig oder im URL Pfad wurde hinter dem Slash kein Wert angegeben ▪ 403: Der Benutzer ist nicht autorisiert für die angegebene Publikation Daten über diesen Endpunkt einzuliefern oder die Publikation wurde nicht zur Bereitstellung über HTTPS konfiguriert.

Tabelle 25: Request/Response zwischen Datengebersystem/Mobilithek beim Publisher Push HTTPS

9.2 Datennehmerseite

9.2.1 Client Pull HTTPS

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

Request an die Mobilithek

Das Datennehmersystem muss einen HTTPS GET-Request an die Mobilithek 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://mobilithek.info:8443/mobilithek/api/V1.0/subscription?subscriptionID=<subscriptionId>
```

Beachten Sie auch die Hinweise in Kapitel 4.8 „Nutzung des „If-Modified-Since“ Header-Feldes“.

Response an das Datennehmersystem

Das Brokersystem der Mobilithek erzeugt nach Erhalt des Requests eine HTTPS Response. Als Statuscodes können die Standard HTTP Statuscodes [HTTP/1.1] auftreten, wobei die in der folgenden Tabelle 26 beschriebenen Bedeutungen gelten. Der Content-Type der Response ist vom gleichen Typ wie durch den Datengeber eingeliefert (in der folgenden Tabelle wird text/csv angenommen) und wird

GZIP-komprimiert versendet. Der Message-Body der Response besteht aus dem angeforderten Datenpaket.

Beschreibung	
Request	Request GET /mobilithek/api/V1.0/subscription?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK Content-Type: text/csv Content-Length: xx spalte1, spalte2, weiterer inhalt
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] Folgende Statuscodes haben eine spezielle Bedeutung: ▪ 204: Kein Datenpaket im Paketpuffer zur Subskription. ▪ 304: Kein Datenpaket im Paketpuffer das jünger als der Zeitstempel im "if-modified-since" header ist. ▪ 400: Kein Subskriptionsparameter, oder numerischer Subskriptionsparameter oder fehlender Accept-Encoding Header ▪ 403: Der Benutzer ist nicht autorisiert diese Subskription über diesen Endpunkt abzurufen bzw. die Publikation kann nicht über diesen Endpunkt bezogen werden ▪ 404: Keine oder eine nicht mehr gültige Subskription zum Subskriptionsparameter gefunden. ▪ 405: Der Parameter ?subscriptionId= wurde nicht spezifiziert bzw. es wurde kein Wert für den Parameter spezifiziert ▪ 406: gzip nicht im "Accept-Encoding" Request Header spezifiziert.

Tabelle 26: Request/Response zwischen Mobilithek/Datennehmersystem beim Client Pull HTTPS

9.2.2 Publisher Push HTTPS

Das Brokersystem der Mobilithek schickt ein Datenpaket zu einer Subskription an ein Datennehmersystem.

Request an das Datennehmersystem

Das Brokersystem der Mobilithek schickt einen HTTPS POST-Request zum Datennehmersystem, in dem die Subskriptions-ID im Header-Element sowie die Nutzdaten im HTTP Request Body übergeben werden. Der Content-Type des Requests ist vom gleichen Typ wie bei der Einlieferung durch den Datengeber spezifiziert (in der folgenden Tabelle wird text/csv angenommen).

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

Die URL muss vom Datennehmer vollständig hinterlegt werden. Die Mobilithek ergänzt diese nicht um Parameter, wie z.B. die Subskriptions-ID.

Response an die Mobilithek

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 27 beschriebenen Bedeutungen gelten.

	Beschreibung
Request	Request POST /datenabgabe HTTP/1.1 Host: datennehmerhost Content-Type : text/csv Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1]

Tabelle 27: Request/Response zwischen Brokersystem der Mobilithek/Datennehmersystem beim Publisher Push HTTPS

Bezugnehmend auf die Ausführungen in Kapitel 4.7.2 führen andere Statuscodes als 200, zu einem wiederholten Übertragungsversuch des Datenpakets.

Ist das Datennehmersystem technisch nicht erreichbar, werden HTTP-Head Request an die hinterlegte URL geschickt. Erst wenn der HTTP-Head Request durch das Datennehmersystem mit einem Status 200 quittiert wird, wird es erneut beliefert.

Das Identity-Encoding wird von der Mobilithek nicht unterstützt.

10 Zertifikatsbasierte M2M-Kommunikation

Die Security-Komponente der Mobilithek 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 an der Administrations-Komponente der Mobilithek durch den Administrator der Organisation beantragt und nach der Beantragung erstellt und dem Organisations-Administrator per E-Mail zugesandt. Das zur Signatur erforderliche Passwort wird per SMS an die hinterlegte Mobilnummer übermittelt.

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

10.1 Aufgaben der Security-Komponente

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

Bevor die an der Mobilithek 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 Mobilithek.

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

Die Vertraulichkeit der Kommunikation zwischen Datengebersystem und Mobilithek einerseits und Mobilithek und Datennehmersystem andererseits, wird durch die ausschließliche Verwendung einer SSL/TLS-Transportverschlüsselung gewährleistet.

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.

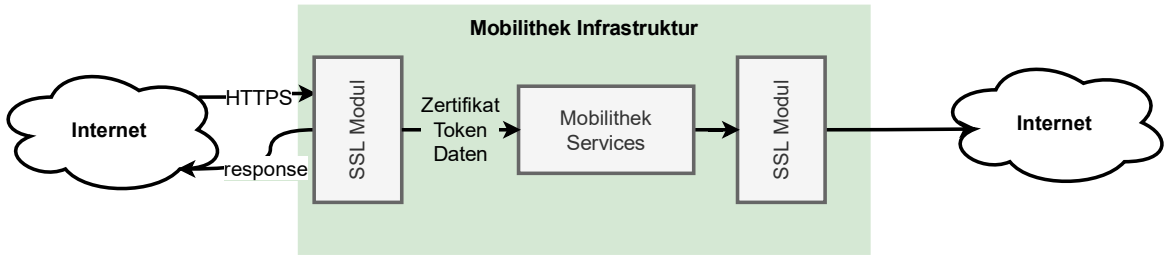

Abbildung 4: Übersicht Sicherheitsarchitektur

Das SSL-Modul 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 Mobilithek weiter.

10.2 Maschinenzertifikat beantragen

Der Betreiber der Mobilithek vermittelt zwischen Datengeber- bzw. Datennehmersystem. Datengeber und Datennehmer beantragen dafür im Rahmen ihrer Registrierung ein oder mehrere Maschinenzertifikate über die Administrations-GUI der Mobilithek. Das Zertifikat wird ihnen anschließend allerdings von der Mobilithek zugesendet.

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

Wie Sie ein Maschinenzertifikat über die Mobilithek beantragen, ist in [FAQ] beschrieben.

10.3 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 11.

10.4 Authentifizierung der Mobilithek als Webclient

Fungiert die Mobilithek 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 Mobilithek abgesetzt wurden.

Die für die Verifikation erforderliche Zertifikatskette kann im Download Bereich (https://mobilithek.info/help/download) der Mobilithek in der Kategorie Zertifikate heruntergeladen werden und muss im Datengeber- bzw. Datennehmersystem hinterlegt werden.

Damit eine SSL Verbindung etabliert werden kann, ist es notwendig, dass auf dem Server der Datengeber- oder Datennehmerseite ein gültiges Server Zertifikat oder ein von der Mobilithek ausgestelltes Maschinenzertifikat installiert ist.

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

10.5 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 Mobilithek 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 von der Mobilithek verwendete Serverzertifikat ist von der Mobilithek signiert. Es ist daher notwendig, dieses im "Truststore" des Systems zu installieren. Das Zertifikat kann aus dem Download Bereich (https://mobilithek.info/help/download) der Mobilithek in der Kategorie Zertifikate heruntergeladen werden.

11 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.example.com.p12 -out ehp.example.com.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:

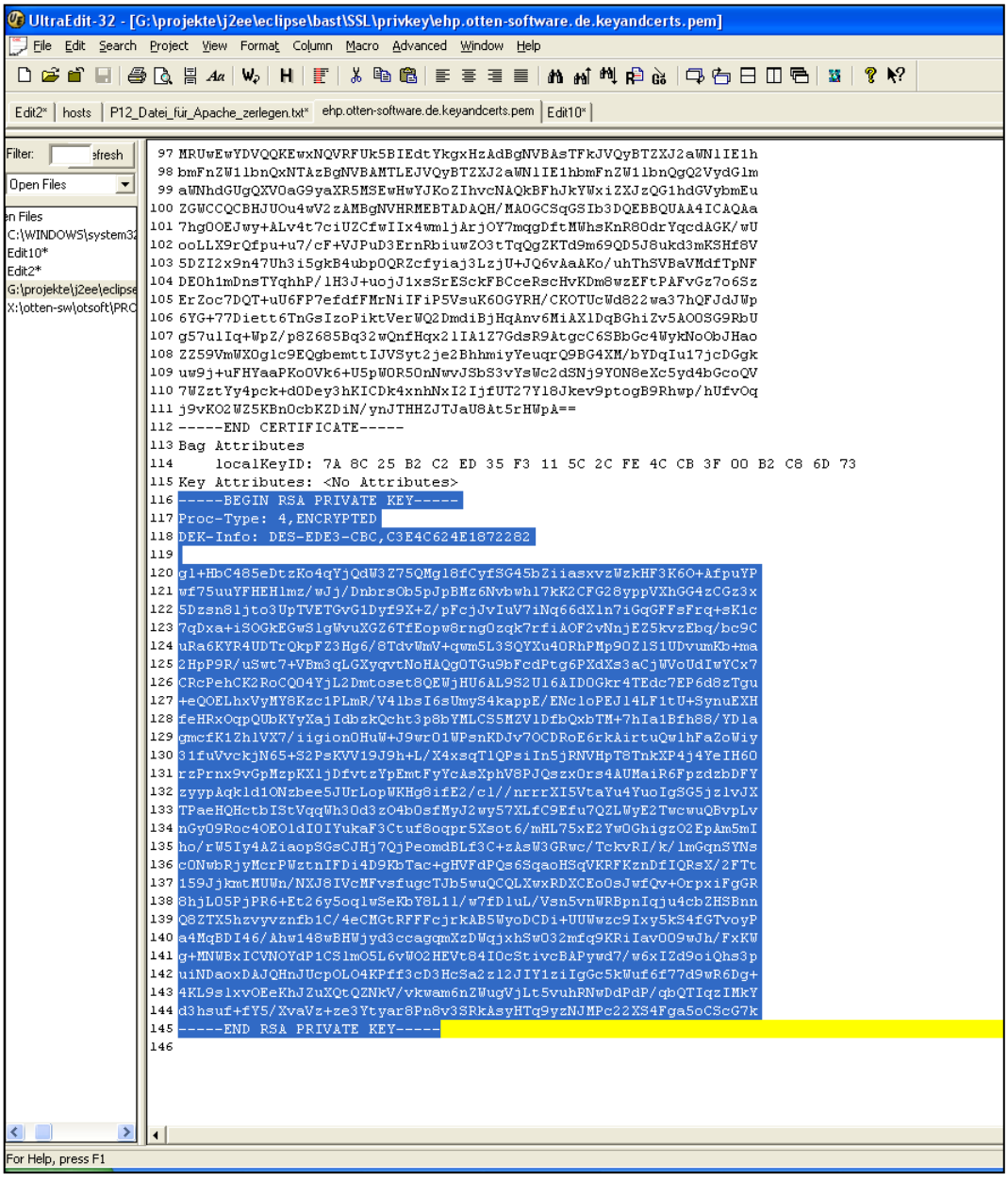

Abbildung 5: Datei <sammeldatei.pem>

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.example.com.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:

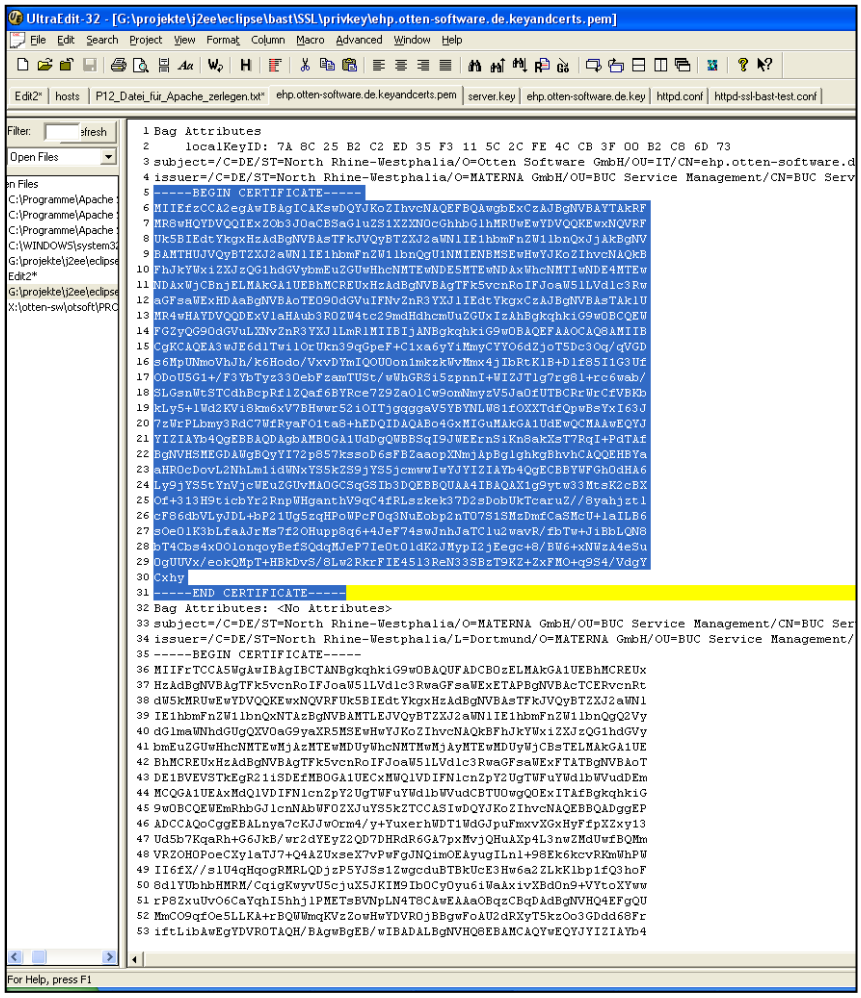

Abbildung 6: Datei <sammeldatei.pem>

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 Mobilithek-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.example.com.crt"
SSLCertificateKeyFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.key\ehp.example.com.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"
```

12 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 MUST 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.	(✓) Abweichung: Clients müssen gzip akzeptieren	(✓) Abweichung: Clients müssen gzip akzeptieren
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.	✗	✗
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	✗	✗

Regel	Verweis auf Regel in [DATEXIIv2PSM], Kapitel 4	Verweis auf Regel in [DATEXIIv3Annex]	DATEX II v2	DATEX II v3
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	✓ 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	✗	✗

13 Anhang C – Änderungsnachweis

Version	Datum	Änderungen
1.2	22.09.2023	• Kleinere Rechtschreibfehler und inhaltliche Fehler korrigiert • Neue Kapitel 6.2.1.2 und 7.3.1.2 für die entsprechenden neuen Schnittstellen hinzugefügt • Unterstützte TLS-Versionen um 1.3 erweitert
1.2.1	04.03.2024	• Behandlung von referenzierten JSON-Subschemas in Kapitel 3 angepasst • Der Accept-Encoding Parameterwert gzip wurde in allen Request-Tabellen von Groß- in Kleinschreibung geändert • Korrektur der Request-URIs in den Kapiteln 6.2.2.1 und 9.2 • XML-Beispiele in Kapitel 8.2.1 wurden korrigiert • Abschnitt zu SNI entfernt, da SNI entgegen dem entfernten Abschnitt von der Mobilithek unterstützt wird • TAG für namespace im ganzen Dokument vereinheitlicht auf <soapenv: > • SOAP (XML)-Rückmeldung in Kapitel 6.1.2.2 angepasst