



Mobilitäts Daten Marktplatz

# Technical Interface Description

Version 3.0.1 – 02.09.2021

## Table of Contents

1	Introduction.....	6
1.1	Preamble .....	6
1.2	Document Structure.....	7
1.3	References .....	7
1.3.1	General.....	7
1.3.2	DATEX II v2.....	8
1.3.3	DATEX II v3.....	8
1.4	List of Abbreviations .....	9
2	Overview of the MDM Platform Components .....	11
3	Data Exchange Formats .....	12
3.1	DATEX II .....	12
3.2	Container Format.....	13
4	Interfaces of the MDM Broker System .....	14
4.1	Communication Encryption .....	15
4.2	Compression .....	16
4.3	Commitment to Invariability.....	16
4.3.1	DATEX II v2.....	16
4.3.2	DATEX II v3.....	16
4.4	Usage of Interfaces.....	18
4.4.1	Data Supplier.....	18
4.4.2	Data Client.....	18
4.5	Usage of the „If-Modified-Since“ Header Field in the HTTPS Protocol .....	19
4.5.1	Data Supplier.....	19
4.5.2	Data Client.....	19
4.5.3	Unchanged Data.....	19
5	DATEX II v2.....	20
5.1	SOAP Interface.....	20
5.1.1	Data Supplier.....	20
5.1.1.1	Client Pull SOAP .....	20
5.1.1.2	Publisher Push SOAP.....	20
5.1.2	Data Client.....	22
5.1.2.1	Client Pull SOAP .....	22
5.1.2.2	Publisher Push SOAP.....	22
5.2	HTTPS Interface.....	24
5.2.1	Data Supplier.....	24
5.2.1.1	Client Pull HTTPS.....	24

5.2.2	Data Client.....	25
5.2.2.1	Client Pull HTTPS.....	25
5.3	OCIT-C Interface .....	27
5.3.1	Features .....	27
5.3.2	Data Supplier – Publisher Push OCIT-C.....	28
5.3.3	Data Client – Client Pull OCIT-C.....	30
5.3.4	Error Handling .....	34
6	DATEX II v3.....	35
6.1	XML Schema Application Notes for Exchange 2020 .....	35
6.1.1	DATEX II v3 Level A or B.....	36
6.1.2	DATEX II v3 Level C.....	36
6.2	SOAP Interface.....	37
6.2.1	Data Supplier.....	37
6.2.1.1	Client Pull SOAP .....	37
6.2.1.2	Publisher Push SOAP.....	39
6.2.2	Data Client.....	41
6.2.2.1	Client Pull SOAP .....	41
6.2.2.2	Publisher Push SOAP.....	43
6.3	HTTPS Interface.....	44
6.3.1	Data Supplier.....	44
6.3.1.1	Client Pull HTTPS.....	44
6.3.2	Data Client.....	45
6.3.2.1	Client Pull HTTPS.....	45
7	Container .....	47
7.1	SOAP Interface.....	47
7.1.1	Data Supplier.....	47
7.1.1.1	Client Pull SOAP .....	47
7.1.1.2	Publisher Push SOAP.....	48
7.1.2	Data Client.....	49
7.1.2.1	Client Pull SOAP .....	49
7.1.2.2	Publisher Push SOAP.....	49
7.2	HTTPS Interface.....	51
7.2.1	Data Supplier.....	51
7.2.1.1	Client Pull HTTPS.....	51
7.2.1.2	Publisher Push HTTPS.....	52
7.2.2	Data Client.....	53
7.2.2.1	Client Pull HTTPS.....	53
7.2.2.2	Publisher Push HTTPS.....	54

8	Certificate-based M2M Communication .....	56
8.1	Tasks of the Security Component .....	56
8.2	Note on Server Name Indication .....	57
8.3	Applying for a Machine Certificate .....	57
8.4	Installing a Machine Certificate and Issuer Certificate .....	57
8.5	Authentication of the MDM Platform as Web Client.....	58
8.6	Authentication of Data Supplier/Data Client Web Clients .....	58
9	Appendix A- Processing the p12 File for Apache Server Configuration.....	59
10	Appendix B – DATEX II HTTP Protocol Support .....	63

## List of Tables

Table 1: References (general) .....	8
Table 2: References (DATEX II v2).....	8
Table 3: References (DATEX II v3).....	9
Table 4: List of abbreviations .....	10
Table 5: Overview of the MDM platform components.....	11
Table 6: Overview of the interfaces of the MDM broker system .....	15
Table 7: MDM operation modes.....	18
Table 8: Request/Response between MDM platform and data client system (Client Pull HTTPS) .....	26
Table 9: OCIT-C error codes used.....	34
Table 10: Request/Response between MDM platform and data client system (Client Pull HTTPS) .....	46
Table 11: Request/Response between data supplier system and MDM platform with client pull HTTPS .....	51
Table 12: Request/Response between data supplier system and MDM platform with publisher push HTTPS.....	53
Table 13: Response between MDM platform/data client system with Client Pull HTTPS .....	54
Table 14: Request/Response between MDM broker system/data client system with Publisher Push HTTPS.....	55

## List of Figures

Figure 1: Components of the MDM platform .....	11
Figure 2: Container Format Overview .....	13
Figure 3: Interfaces between data provider, broker system and data client .....	14
Figure 4: Overview of the security architecture .....	57
Figure 5: Datei <sammeldatei.pem> .....	60
Figure 6: File <sammeldatei.pem>.....	61

# **1 Introduction**

## **1.1 Preamble**

The Mobility Data Marketplace (MDM) aims at supporting the exchange of data between data suppliers and data clients using interfaces. At the same time, it is a central portal with collected information about available online traffic data of individual data suppliers. Thus, the MDM platform allows its users to offer, find and subscribe to online traffic-related data without the necessity of any time-consuming search for relevant data and a complex technical and organizational coordination between data clients and data suppliers. The data exchange is handled via standardized interfaces. In conclusion, the business processes should be simplified for all parties involved and the potential of existing data sources should be exploited.

This interface description is aimed at potential data suppliers and data clients. [SOAP] It is presupposed that knowledge in the implementation and operation of SOAP web services or HTTPS client/server architectures are provided to use the interfaces of the MDM system.

The data transfer between the MDM platform and the data supplier or data client systems can be supplied via SOAP-based web services or simple HTTPS-GET/POST requests. In addition, the transmission by OCIT-C protocol is provided.

## 1.2 Document Structure

This document consists of the following sections:

- Section 1 provides a brief overview, the referenced documents and the list of abbreviations
- Section 2 describes the components of the MDM system.
- Section 3 handles the available data formats.
- Section 4 describes the interfaces of the MDM platform for M2M communication.
- Section 5 describes the DATEX II v2 format in detail.
- Section 6 describes DATEX II v3 accordingly.
- Section 7 describes the proprietary MDM container format.
- Section 8 describes the measures which secure the M2M communication.

## 1.3 References

### 1.3.1 General

[Source]	Publisher
[BHB]	MDM User Manual, V3.0.0 <a href="https://service.mdm-portal.de/doc/MDM-UserManual.pdf">https://service.mdm-portal.de/doc/MDM-UserManual.pdf</a>
[GZIP]	RFC 1952 (Mai 1996) GZIP File Format Specification Version 4.3, <a href="https://tools.ietf.org/rfc/rfc1952.txt">https://tools.ietf.org/rfc/rfc1952.txt</a>
[HTTP/1.1]	RFC 2616 (Juni 1999) Hypertext Transfer Protocol -- HTTP/1.1 <a href="https://www.ietf.org/rfc/rfc2616.txt">https://www.ietf.org/rfc/rfc2616.txt</a>
[HTTPS]	RFC 2818 (Mai 2000) HTTP over TLS <a href="https://www.ietf.org/rfc/rfc2818.txt">https://www.ietf.org/rfc/rfc2818.txt</a>
[MCS]	MDM Container Format Specification <a href="https://www.mdm-portal.de/wp-content/uploads/2019/03/mdm_spezifikation_container-v1.1.pdf">https://www.mdm-portal.de/wp-content/uploads/2019/03/mdm_spezifikation_container-v1.1.pdf</a>
[OCIT-C]	OCIT-C Specification Version 1.1_R1 vom 30.10.2014 <a href="https://www.ocit.org/media/ocit-c_protokoll_v1.1_r1.pdf">https://www.ocit.org/media/ocit-c_protokoll_v1.1_r1.pdf</a> <a href="https://www.mdm-portal.de/files/ocit.wSDL">https://www.mdm-portal.de/files/ocit.wSDL</a>
[PKI]	RFC 2459 (January 1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile <a href="https://www.ietf.org/rfc/rfc2459.txt">https://www.ietf.org/rfc/rfc2459.txt</a>
[SOAP]	SOAP Version 1.2 <a href="https://www.w3.org/TR/soap12-part1/">https://www.w3.org/TR/soap12-part1/</a>

[Source]	Publisher
[URL]	RFC 1738 (Dezember 1994) Uniform Resource Locators (URL) <a href="https://www.ietf.org/rfc/rfc1738.txt">https://www.ietf.org/rfc/rfc1738.txt</a>
[X.509v3]	ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection – The Directory: Authentication Framework, June 1997. <a href="https://www.itu.int/rec/T-REC-X.509-199708-S/en">https://www.itu.int/rec/T-REC-X.509-199708-S/en</a>

Table 1: References (general)

### 1.3.2 DATEX II v2

[Source]	Publisher
[DATEXIIv2PSM]	DATEX II v2.0 Exchange Platform Specific Model
[DATEXIIv2Pull]	DATEX II v2.0 Pull wsdl <a href="https://www.mdm-portal.de/files/Pull.wsdl">https://www.mdm-portal.de/files/Pull.wsdl</a>
[DATEXIIv2Push]	DATEX II v2.0 Push wsdl <a href="https://www.mdm-portal.de/files/Push.wsdl">https://www.mdm-portal.de/files/Push.wsdl</a>
[DATEXIIv2Schema]	DATEX II XML Schema 2.0
[DATEXIIv2SDG]	DATEX II v2.0 Software Developers Guide, Version v.1.2
[DATEXIIv2Spec]	Includes the following documents, which are available to all registered users for download under <a href="https://www.datex2.eu">https://www.datex2.eu</a> : [DATEXIIv2PSM], [DATEXIIv2UG]
[DATEXIIv2UG]	DATEX II v2.0 User Guide v.1.2

Table 2: References (DATEX II v2)

### 1.3.3 DATEX II v3

[Source]	Publisher
[DATEXIIv3Annex]	Annexes to Platform Specific Model: <a href="https://docs.datex2.eu/exchange/2020/psm/annexes.html">https://docs.datex2.eu/exchange/2020/psm/annexes.html</a>
[DATEXIIv3Pull]	DATEX II v3.0 Snapshot Pull wsdl <a href="https://www.mdm-portal.de/files/SnapshotPull.wsdl">https://www.mdm-portal.de/files/SnapshotPull.wsdl</a>
[DATEXIIv3Push]	DATEX II v3.0 Snapshot Push wsdl <a href="https://www.mdm-portal.de/files/SnapshotPush.wsdl">https://www.mdm-portal.de/files/SnapshotPush.wsdl</a>
[DATEXIIv3Exc]	Download for Level A and Level B publications: <a href="https://www.mdm-portal.de/files/Exchange2020LevelAB.zip">https://www.mdm-portal.de/files/Exchange2020LevelAB.zip</a> Download for Level C publications: <a href="https://www.mdm-portal.de/files/Exchange2020LevelC.zip">https://www.mdm-portal.de/files/Exchange2020LevelC.zip</a>



[Source]	Publisher
[DATEXIIv3ExcUG]	Exchange 2020 User Guide : <a href="https://docs.datex2.eu/exchange/2020/userguide/">https://docs.datex2.eu/exchange/2020/userguide/</a>

Table 3: References (DATEX II v3)

## 1.4 List of Abbreviations

Abbreviation	Resolution
BASE64	BASE64 describes a method of encoding 8-bit binary data into a string that consists only of readable code page-independent ASCII characters.
BASt	Bundesanstalt für Straßenwesen (Federal Highway Research Institute)
DE	German
GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IIS	Microsoft Internet Information Services
JSSE	Java Secure Socket Extension
M2M	Machine-to-Machine
MDM	Mobility Data Marketplace
MDV	Metadatenverzeichnis (metadata directory)
OCIT	Open Communication Interface for Road Traffic Control Systems
PAS	Publicly Available Specification
PKI	Public Key Infrastructure
PSM	Platform Specific Model
RC	Release Candidate
RFC	Request for Comments
SDG	Software Developers Guide
SNI	Server Name Indication
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator
UTF	UCS Transformation Format

Abbreviation	Resolution
WS	Webserver
WSDL	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

*Table 4: List of abbreviations*

## 2

## Overview of the MDM Platform Components

The MDM platform consists of four components that fulfill different roles.

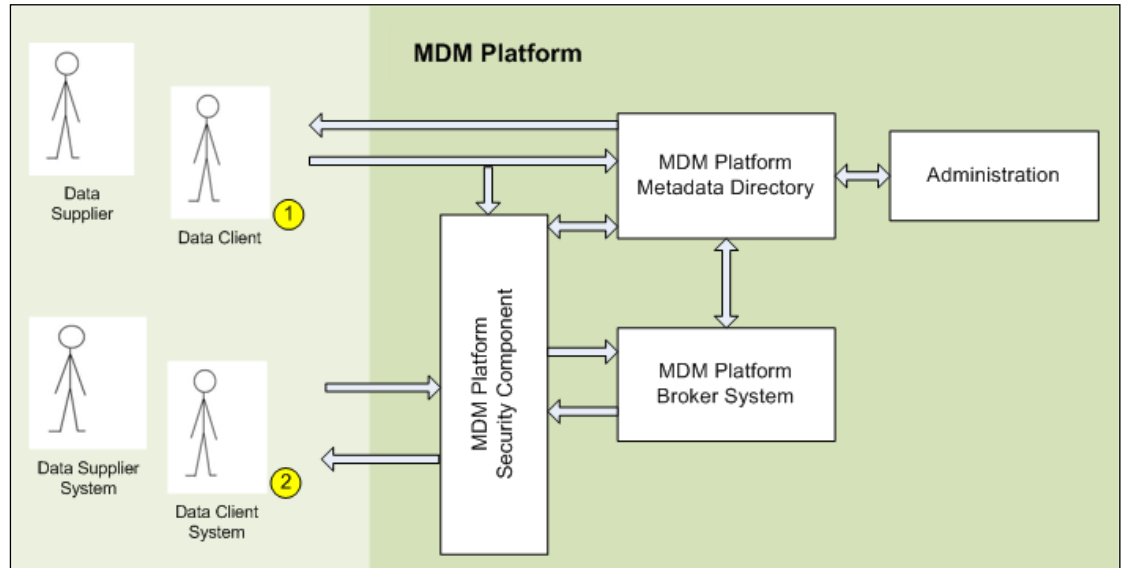


Figure 1: Components of the MDM platform

Component	Description
Security component	Via the security component, the data client system/data supplier system can be authenticated to use the services.
Metadata directory	The metadata directory is used to manage all information relevant to MDM platform and provides a number of organizational services.
Broker system	The broker system handles the actual processing of the data packets and it is, therefore, the focus of this interface description.
Administration	The administration is realized by means of a web-based user interface (GUI), see [BHB]

Table 5: Overview of the MDM platform components

The following communication and application scenarios are supported by the MDM platform:

- Interested parties as well as data clients and data suppliers can communicate with the metadata directory by using the web GUI, to access services, such as researching or registering. To view or edit certain content of the metadata directory, an authentication must first be run throughout the MDM platform security component.
- After authentication via the security component, the data client and data supplier systems can establish an M2M communication with the broker system to deliver or request data.

### 3 Data Exchange Formats

To exchange mobility data between the broker system and the data supplier / data client systems, the following data formats are specified:

- To allow the use of the platform by standard compliant DATEX II implementations at data suppliers or data clients, the MDM platform supports the format DATEX II, which is based on XML, by using native interfaces.
- To create an independent generic interface from specific formats, a new data format is provided for transmission. It refers to the so-called container format that can be transmitted over the arbitrary XML and binary data.

The validity of data delivered at the MDM broker interface is not checked automatically. Instead, the MDM web GUI provides a button to validate the currently stored data packet of a publication. Therefore, the data schemas are retrieved via the URL specified in the profile of the publication.

For publications in DATEX II format, it is the responsibility of the data supplier to provide the correct file schema. For publications in container format, the standard schema is already made available under a generally valid URL. Please, reference this URL in the "schemaLocation" attribute of your XML data packets to provide data clients with an automatic validation of the packets. The MDM accepts the data packets independent of the validation result and delivers them to the data clients even if the result is negative.

#### 3.1 DATEX II

DATEX II is a European standard for exchanging mobility data. Basic knowledge of DATEX II specification is required for this section [DATEXIIv2Spec]. The original implementation of the MDM platform is based on DATEX II v2. Currently, v2 and v3 are supported.

DATEX II defines XML structures for the exchange of mobility data. The underlying schema can be viewed under <https://www.datex2.eu/>. The payload must be defined based on this schema. DATEX II determines not only a standard for the structure of the payload, but also regulates the exchange process. The latter is described in detail in chapter 4.

The underlying documents on which the DATEX II is based are listed in chapter 1.3 References as [DATEXIIv2Spec]. The structure of the DATEX II payload is not relevant to the MDM platform, as the latter transfers the data unchanged and does not process the data in any way.

DATEX II not only provides for the dispatch of complete data packets, but also for sending updates to previous versions. This DATEX II option is **not** supported by the MDM platform: Both the data supplier system and the MDM broker system must always send complete data sets.

This means that each packet contains all records of the relevant publication that are known to the data supplier. These records are valid at the time of packet sending. It is therefore not possible to send only changes to the "last known" state. This may seem to be a disadvantage, but it is a requirement that is essential to the preservation of the MDM system scalability. The disadvantage of this partial redundancy is tacitly accepted, as it is considered by the scalable architecture of

the platform and the performance of modern ICT infrastructure. It should be borne in mind that the MDM platform diminishes the burden of scalability of the data suppliers.

## 3.2 Container Format

In addition to the DATEX II standard mentioned in the previous section, another XML-based model for the transmission of data is supported by the MDM platform. This container format called data format has been specially created for the exchange of data via the MDM. The schema of the data format is found in the container format specification [MCS]. In addition to the actual payload that is contained in a body element, the data format allows more structural information to be transmitted in a header element. This information is particularly used to control the communication process.

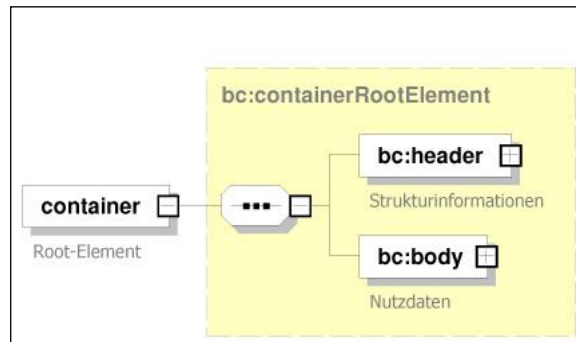


Figure 2: Container Format Overview

To keep the model flexible, the format and content of the body element is not specified. Thus, not only data in XML format can be transported in containers, but also binary data.

## 4 Interfaces of the MDM Broker System

The MDM broker system takes the role of the client or the role of the server as an intermediary between the data supplier system and the data client system, depending on the situation:

- As a client, the broker system can request data from the data supplier or the data supplier can - on his own initiative - send the data to the broker system.
- As a client, the data client can on its part request data from the broker system or the broker system can send - on its own initiative - the data to the data client.

Figure 3 shows the possible paths that are available for data packet transmission between the data supplier and the broker system on the one hand and the broker system and the data client on the other.

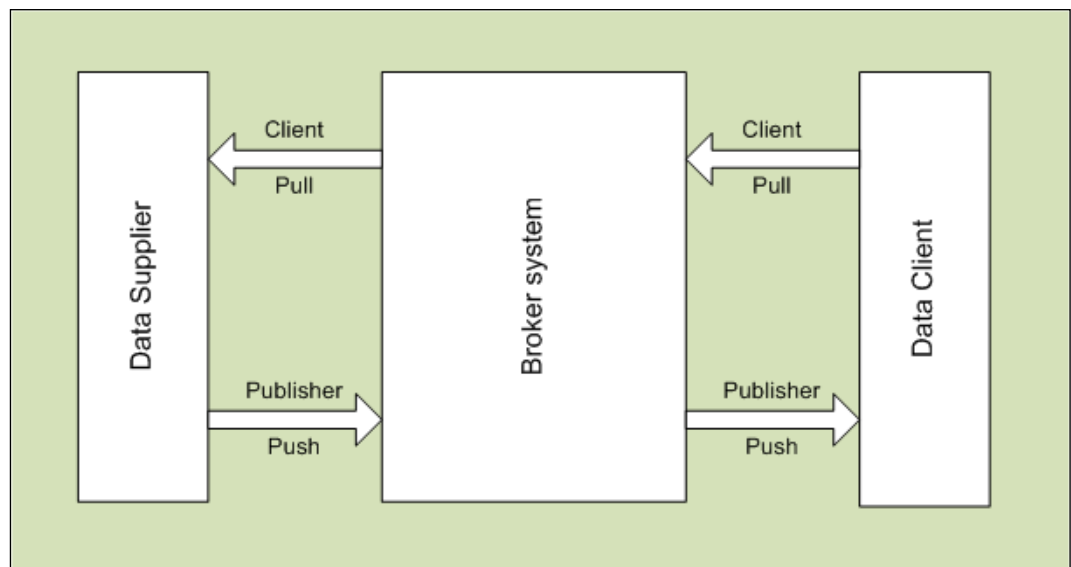


Figure 3: Interfaces between data provider, broker system and data client

The data packets received or sent by the broker system must have DATEX II format or the proprietary container format.

The transmission protocols HTTPS and SOAP via HTTPS are supported for each format. For the format DATEX II, the OCIT-C protocol is also supported.

Table 6 shows what communications are supported. The section in which the relevant communication is described - distinguished by the data supplier and data client systems - is mentioned for each data format (DATEX II / Container), communication pattern (Client Pull / Publisher Push) and protocol (HTTPS, SOAP, OCIT-C), if supported.

It is additionally indicated whether the data supplier or data client system acts as a client or as a server towards the MDM. Client here means that the system makes enquiries to the MDM or actively establishes the connection to it.

On the other hand, server means that the system is contacted by the MDM and must answer its enquiries. In this case, the MDM must be granted external network access to the system to be connected.

		Data Supplier System			Data Client System		
		SOAP	HTTPS	OCIT	SOAP	HTTPS	OCIT
<b>DATEX II v2</b>	Client Pull	<a href="#">5.1.1.1</a> Server	<a href="#">5.2.1.1</a> Server	-	<a href="#">5.1.2.1</a> Client	<a href="#">5.2.2.1</a> Client	<a href="#">5.3.3</a> Client
	Publisher Push	<a href="#">5.1.1.2</a> Client	-	<a href="#">0</a> Client	<a href="#">5.1.2.2</a> Server	-	-
<b>DATEX II v3</b>	Client Pull	<a href="#">6.2.1.1</a> Server	<a href="#">6.3.1.1</a> Server	-	<a href="#">6.2.2.1</a> Client	<a href="#">6.3.2.1</a> Client	-
	Publisher Push	<a href="#">6.2.1.2</a> Client	-	-	<a href="#">6.2.2.2</a> Server	-	-
<b>Container</b>	Client Pull	<a href="#">7.1.1.1</a> Server	<a href="#">7.2.1.1</a> Server	-	<a href="#">7.1.2.1</a> Client	<a href="#">7.2.2.1</a> Client	-
	Publisher Push	<a href="#">7.1.1.2</a> Client	<a href="#">7.2.1.2</a> Client	-	<a href="#">7.1.2.2</a> Server	<a href="#">7.2.2.2</a> Server	-

Table 6: Overview of the interfaces of the MDM broker system

If the SOAP method is used, the WSDL of the broker service can generally be queried at the service endpoint that is specific to the relevant publication or subscription using the "?wsdl" request.

Generally, a data packet can only be retrieved by data clients for the time it is valid as specified in the publication profile. With expired validity, the packet is deleted from the buffer.

For the time no new data packet is available, the data client system receives a „No Content“ error notification which differs for each protocol. Details are provided in the protocol descriptions.

## 4.1 Communication Encryption

The interfaces offered by the MDM platform can be used by the data supplier systems and data client systems using the services of the platform. These services for data collection or deliveries are provided by using defined and unified URLs [URL] and require a certificate-based client authentication via HTTPS [HTTPS]. For this client authentication, X.509-compliant certificates are used [PKI] which are issued by the operator of the MDM platform.

## 4.2 Compression

When transmitting data between the MDM platform and the data supplier systems, both GZIP-encoded (i.e., compressed) and uncompressed HTTPS requests and responses are supported.

Data between the MDM platform and the data client systems is always transmitted – differing from what is stated in [DATEXIIv2PSM] – by means of GZIP-encoded HTTPS requests and responses.

This always applies, no matter which Exchange protocol is used for HTTP, SOAP and OCIT-C.

## 4.3 Commitment to Invariability

The MDM platform has been designed to forward any data from data suppliers to data clients without modifications. The DATEX II payload, i.e. the included data packages, must not be changed by the broker system.

For this principle, "Commitment to Invariability" is the established term.

When the data is supplied via SOAP, this principle is extended to the invariability of the SOAP envelope.

As a major implication of the "Commitment to Invariability", any namespace declaration for the DATEX II payload has to be included in the <d2LogicalModel> element (for DATEX II v2) or in the so-called messageContainer (for DATEX II v3) so that these declarations remain a part of the payload delivery if, e.g., the data is supplied via SOAP and then forwarded via HTTP (in this case, the SOAP envelope is removed).

An example each for DATEX II v2 and DATEX II v3, respectively, is provided next where the "Commitment to Invariability" is considered properly.

### 4.3.1 DATEX II v2

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

### 4.3.2 DATEX II v3

```
<?xml version='1.0' encoding='UTF-8'?>
<S:Body>
  <con:messageContainer xmlns:con="http://datex2.eu/schema/3/messageContainer"
```



```

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

```

## i

## Important notes on DATEX II v3

1. The mandatory codedExchangeProtocol element depends on the applied receipt / delivery protocol.
2. With each data delivery from the MDM to the data client system, the value of this element is as set follows:
  - a. „snapshotPush“, if the MDM delivers the package to the data client via SOAP-Push
  - b. „snapshotPull“, if the data client sends a SOAP Pull request to the MDM
  - c. „snapshotPull“, if the data client sends an HTTP Pull request to the MDM
3. The value of the messageGenerationTimestamp element is not changed when data is provided to the data client system by the MDM, i.e. the timestamp is an end-to-end value, which equals the original timestamp of the provided data package.

## 4.4 Usage of Interfaces

When using the HTTPS or SOAP protocol, there are three different modes of operation for the exchange of data, all of which are supported by the MDM platform:

Mode	Description
Client Pull	The communication is initiated by the client (MDM broker system to data supplier or data client system to MDM platform) and the data is sent as a response.
Publisher Push Periodic	The communication is initiated by the publisher (data supplier system to MDM platform) at timed intervals.
Publisher Push on Occurrence	The communication is always initiated by the publisher (data supplier system to MDM platform or MDM broker system to data client) if the data changes.

*Table 7: MDM operation modes*

For data exchange with the MDM, transport encryption with TLS 1.2 and authentication by means of standard compliant X.509v3 certificates have to be used. If the standard protocols provide for basic authentication by user name and password, any of these protocol elements will be ignored. This applies to the OCIT-C protocol [OCIT-C] in particular, as described in the following.

The MDM uses an OCIT-C interface based on the OCIT-C standard in version 1.1\_R1 from 30/10/2014. The OCIT-C functionality is restricted by the MDM and is provided under the stipulation of a specific use of protocol elements. OCIT-C uses only DATEX II v2 as data model, as described in this document or in [DATEXIIv2Spec]. OCIT-C data models are not supported.

**Note:** As of January 01, 2021, the OTS 2 protocol is no longer supported.

### 4.4.1 Data Supplier

Towards the data supplier (the publisher), the MDM broker system appears as a subscriber who receives the data packets. The broker system can assume the role of a server or a client, depending on the procedure.

When using the OCIT-C protocol, the broker system acts as a server and the data supplier system acts as a client.

### 4.4.2 Data Client

Towards the data client (the subscriber), the MDM broker system appears as a publisher who provides the data packets. The broker system can assume the role of a server or a client, depending on the procedure.

When using the OCIT-C protocol, the broker system acts as a server and the data client system as a client.

## 4.5 Usage of the „If-Modified-Since“ Header Field in the HTTPS Protocol

The broker system supports the "If-Modified-Since" in association with the "Last-Modified" header field (see [HTTP/1.1]). As a result, the transfer of already collected data packets can be prevented.

### Example:

If the response of the previous data packet contains the following header line

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

the next data packet will be requested with a request, which contains the following header line:

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

### 4.5.1 Data Supplier

The broker system sends the request with an "If-Modified-Since" header field whenever the data supplier system had set the header field "Last-Modified" in its response.

The data supplier system should always set this header field to enable the MDM platform to use this feature.

### 4.5.2 Data Client

The responses of the broker system always contain the header field "Last-Modified". If the data client system wants to use this feature, it must always transmit the value from the last Last-Modified header field.

It is strongly recommended to implement this feature on the data client side!

### 4.5.3 Unchanged Data

If a DATEX II client pull request uses the "If-Modified-Since" header field, and if there are no more recent data packets than those already gathered, an HTTP status code 304 = "Not-Modified" will be generated.

## **5 DATEX II v2**

### **5.1 SOAP Interface**

#### **5.1.1 Data Supplier**

##### **5.1.1.1 Client Pull SOAP**

As with the Client Pull SOAP exchange process, the MDM broker system requests the data supplier system to deliver its data to the MDM platform.

##### **5.1.1.1.1 Offering a Web Service**

The data supplier system must provide a web service with the method `getDatex2Data` that is defined according to the DATEX II Pull WSDL [DATEXIIv2Pull]. Null is thereby expected as input. As output, the MDM broker system receives the requested data in DATEX II format. In the body element, an object of the `d2LogicalModel` type is expected.

Via the MDM administration component, the data supplier must enter its URL in the publication configuration.

##### **5.1.1.1.2 Calling up a Web Service**

The MDM broker system has to provide a web service client that is defined according to the DATEX II pull WSDL [DATEXIIv2Pull] to invoke web services. This web service must return data according to the schema [DATEXIIv2Schema]. A suitable profile from the entire schema is expected to be used.

The broker system identifies the data supplier systems that have subscribed to a pull method and the associated service endpoints in the metadata directory and periodically calls them up according to the configured publication frequency. The data received after the call is cached in corresponding packet buffers to be provided to potential data clients. A previous data packet, if it still exists, will be replaced.

##### **5.1.1.2 Publisher Push SOAP**

With the Publisher Push exchange process, the data supplier system must deliver the data to the MDM platform on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is irrelevant to the operation of the MDM broker system. The mechanism for the exchange is the same in both cases.

##### **5.1.1.2.1 Offering a Web Service**

The MDM broker system provides a web service with the method `putDatex2Data` that is defined based on the specification DATEX II Push WSDL [DATEXIIv2Push]. The data to be supplied is expected as input. As output, the data supplier system receives confirmation data in DATEX II format. In the body element, respectively, an object of the `d2LogicalModel` type is expected.

The output consists of an acknowledgement of receipt.

In the service endpoint URL of the broker system, the ID of the target publication for the data packets is entered.

The URL is structured as follows:

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

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

#### 5.1.1.2.2 Calling up the Web Service

The data supplier system has to provide a web service client that is defined according to DATEX II Push WSDL [DATEXIIv2Push] to call up the web service. The web service must deliver the data to the publication-specific service endpoint of the MDM broker system. The URL of the service endpoint to be used is displayed in the MDM portal publication configuration. The MDM broker system accepts this data and stores it in a packet buffer. A previous data packet, if it still exists, will be replaced.

## 5.1.2 Data Client

### 5.1.2.1 Client Pull SOAP

With the Client Pull SOAP exchange process, the data client system must prompt the MDM platform to transfer the data to the data client system.

#### 5.1.2.1.1 Offering a Web Service

The MDM broker system provides a web service with the method putDatex2Data that is defined based on the specification [DATEXIIv2Pull]. As input, the subscription ID is expected here in the URL, as output, the data client receives the requested data in DATEX II format. In the body element, an object of the d2LogicalModel type is expected. Based on the transmitted subscription ID, the MDM platform can find the corresponding packet buffer and the data packet.

**Example:**

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

#### 5.1.2.1.2 Calling up the Web Service

The data client system must provide a web service client that is defined according to the specification [DATEXIIv2Pull] to invoke web services. The corresponding subscription ID must be carried in the URL as input parameter.

The SOAP endpoint of the broker system is as follows:

```
https://broker.mdm-portal.de/BASSt-MDM-Interface/srv/<subscription
ID>/clientPullService
```

### 5.1.2.2 Publisher Push SOAP

With the Publisher Push exchange process, the MDM broker system delivers the data to the data client systems on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is in this case irrelevant; the mechanism for the delivery to the data client is identical.

#### 5.1.2.2.1 Offering a Web Service

The data client system must provide a web service with the method putDatex2Data that is defined according to the specification [DATEXIIv2Push]. The data to be supplied is expected as input. As output, the MDM platform receives confirmation data in DATEX II format. In the body element, respectively, an object of the d2LogicalModel type is expected. The format of the input parameter corresponds to the DATEX II schema [DATEXIIv2Schema].

#### 5.1.2.2.2 Calling up the Web Service

The MDM broker system provides a web service client that is defined according to [DATEXIIv2Push] to invoke the web services of the data client system. Via the MDM administration component, the data client must enter its service endpoint in the subscription configuration.

The broker system identifies the data client systems and launches a corresponding web service call.

If the data transfer could be successfully completed, the broker system would then expect a confirmation message from the data client system. The following example shows the content of a so-called acknowledge response, which is included in the body element of a submission via SOAP protocol:

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

## **5.2 HTTPS Interface**

### **5.2.1 Data Supplier**

#### **5.2.1.1 Client Pull HTTPS**

As with the client pull exchange process, the MDM broker system periodically requests the data supplier system to deliver its data to the MDM platform. The time interval used must be configured in the metadata directory when configuring the data services. For this exchange, items C.1-C.12 from the Simple HTTP Server Profile of [DATEXIIv2PSM], chapter 4, shall apply.

It should be noted that the additional, optional rules do not apply. The options for authentication (C.13, C.14, C.17) do not apply, as they are obsolete when using the HTTPS method that is compulsory for MDM. C.18-C.27 do no longer apply, since the options relate only to the optional provision of DATEX II data in file format, which is not applicable to MDM. See also Appendix B – DATEX II HTTP Protocol Support.

##### **5.2.1.1.1 Request to Data Supplier**

The MDM broker system sends an HTTPS GET request to the data supplier system from which the data is to be collected. The MDM platform is able to identify the data supplier systems that have subscribed to a pull method, and to send requests to them at defined intervals.

Via the MDM administration component, the data provider must enter the publication-specific server URL in the publication configuration.

Also, consider chapter 4.5, Usage of the „If-Modified-Since“ Header Field.



#### 5.2.1.1.2 Response to the MDM Platform

After receipt of the request, the data supplier system must generate an HTTPS response whose message body consists of the requested DATEX II data. Pursuant to [DATEXIIv2PSM] section 4, the response has the content type "text/xml; charset=utf-8" and can be available as GZIP encoding.

The MDM broker system accepts this data and stores it in a packet buffer. A previous data packet, if it still exists, will be replaced.

### 5.2.2 Data Client

#### 5.2.2.1 Client Pull HTTPS

With the client pull exchange process, the data client system must prompt the MDM broker system to transfer the data.

##### 5.2.2.1.1 Request to the MDM Platform

The data client system must send an HTTPS GET request to the URL of the MDM platform. Due to the subscription ID, the associated packet buffer and the data packet are determined.

The subscription ID has to be provided in the URL path and as parameter. The URL of the broker system is therefore constructed as follows:

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

Also, consider chapter 4.5, Usage of the „If-Modified-Since“ Header Field.

#### 5.2.2.1.2 Response to Data Client

The MDM broker system generates an HTTPS response after receipt of the request. For this purpose, the associated packet buffer and the appropriate data package will be determined based on the subscription ID. The content of the data packet is sent to the data client in the body of the response. Pursuant to DATEX II Client Pull HTTP profile [DATEXIIv2PSM] section 4, the response has the content type „text/xml; charset=utf-8“ and is always sent GZIP compressed, deviating from the specification in [DATEXIIv2PSM].

Default HTTP status codes may occur according to [HTTP/1.1], while their significance is given in Table 8.

Description	
Request	Request GET /BASt-MDM-Interface/srv/clientPullService?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx < d2LogicalModel > ... </d2LogicalModel >
Status codes	Standard HTTP1.1 Status codes [HTTP/1.1] The following status codes have special significance: - 304: not modified, the requested resource is not transmitted again - 403: authentication error - 404: no data packet in packet buffer of the subscription, no subscription or invalid subscription found for subscription parameter - 503: service unavailable (e.g., in case of maintenance)

Table 8: Request/Response between MDM platform and data client system (Client Pull HTTPS)

## 5.3 OCIT-C Interface

**Note:** This interface only supports DATEX II v2. It does not support the DATEX II v3 standard!

### 5.3.1 Features

From the functionality of the OCIT-C standard, the MDM implements the subset of protocol functions needed for the transmission of the current data packet with all the information of a publication. According to the completeness paradigm of MDM, the exchange of data subsets (delta supplies) is not supported. Historical data cannot be queried neither.

The MDM implements a web service with the complete WSDL OCIT\_Cif.wsdl, which is accessible under the specific OCIT context <https://brokermdm-portal.de/BASt-MDM-OCIT-Interface/ocit/>. The call for an unsupported action is, however, acknowledged with a SOAP fault with the value "action not supported".

The data schema is defined by the OCIT-C schema protokoll.xsd. To transport the data, the OCIT messages use a data list, which can contain multiple data objects. In communicating with the MDM, the data list must always contain only one data object. The DATEX II packet has to be transparently embedded into the <data> element of the message. Data submissions with multiple packets are acknowledged with an error.

The <data> element of the OCIT-C message is specified in the protokoll.xsd as an element of type anyType. For a SOAP-compliant transmission, the <data> element has to be typed. For this purpose, a new data type anyD2LogicalModel is introduced using the following OcitCDateX2.xsd.

```
<?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>
```

The schema can be referenced using the following URL:  
<http://bast.s3.amazonaws.com/schema/1446644360562/OcitCDateX2.xsd>

### 5.3.2 Data Supplier – Publisher Push OCIT-C

The Publisher Push functionality is represented by the OCIT-C method “put”. A put call must always be assigned uniquely to a publication by referencing a publication ID. This publication ID that is automatically assigned by the MDV of MDM has to be transmitted by the data supplier system in OCIT-C element <objectType>.

A put message must contain exactly one element of the DATEX II type D2LogicalModel. For this purpose, the request has to contain a data list with exactly one data object. A call with more data objects will be rejected by the MDM with an error. The delivery of a DATEX II element must always be complete, i.e., it must include all data points or objects of the publication. However, this cannot be checked by MDM. It is the responsibility of the data supplier system to ensure this completeness.

In the MDM metadata administration, the DATEX II element can be manually validated against the publication schema stored in the MDM. This schema may only describe the DATEX II payload without the OCIT-C container. The OCIT message is not fully validated.

The following subsection is an example of a possible delivery in OCIT-C format for a publication with the fictitious ID=2600103 of a fictitious organization „TEST“. The DATEX II payload is shown in abbreviated form.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="https://www.w3.org/2001/XMLSchema"
  xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <put xmlns="https://odg_und_partner/OCIT_C">
      <userName>Hello</userName>
      <passWord/>
      <objectType>2600103</objectType>
      <putList>
        <putds>
          <identifier>
            <ident>test</ident>
          </identifier>
          <data xsi:type="ns1:anyD2LogicalModel"
            xmlns:ns1="https://odg_und_partner/OCIT_C/DateX"
            xsi:schemaLocation="https://bast.s3.amazonaws.com/schema/1446644360562/OcitCDateX2.xsd">
            <ns2:d2LogicalModel modelBaseVersion="2" extensionName="MDM"
              extensionVersion="00-01-03"
              xmlns:ns2="https://dateX2.eu/schema/2/2_0"
              xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="https://bast.s3.amazonaws.com/schema/1370477853100/MDM-Profile_ParkingFacilityStatus.xsd">
              <ns2:exchange>
                <ns2:supplierIdentification>
                  <ns2:country>de</ns2:country>
                  <ns2:nationalIdentifier>DE-MDM-TEST</ns2:nationalIdentifier>
                </ns2:supplierIdentification>
              </ns2:exchange>
            </ns2:d2LogicalModel>
          </data>
        </putds>
      </putList>
    </put>
  </soapenv:Body>
</soapenv:Envelope>
```

```

        <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>

```

When data is delivered, the MDM ignores the following items in the OCIT-C protocol:

- username
- password
- identifier within the “putds” attribute

The MDM confirms the delivery with an OCIT message of putResponse type. The elements are set as follows:

- lastStart = Time of delivery
- errorCode = 0; Basically, a formally correct delivery is always acknowledged as error-free, regardless of the quality of the data packet.
- errorText = without content
- badList = empty element

The following subsection shows a sample response.

```

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

```

### 5.3.3 Data Client – Client Pull OCIT-C

The Client Pull functionality is represented by the following three OCIT-C methods:

- inquireAll
- get
- wait4Get

After its launch with the inquireAll method, an OCIT-C client can synchronize with the current data state. For this purpose, the MDM supports the inquireAll method. In the inquireAllResponse, the MDM passes the last valid packet and its internal MDM-ID over to the client. Subsequently, the client can collect the current packages on an ongoing basis, using the methods get or wait4Get. Here, the client must refer to its last packet ID. If no new packet is available in the MDM, the get method will immediately return with an empty response. The wait4Get method will wait until a current data packet is available or a maximum timeout, which is predetermined by the client or defined by the server, has been reached. By using the wait4Get method, a quasi push feature can be implemented toward the data client. In contrast to the actual OCIT-C behavior, the MDM always returns a full data packet with a get- or wait4GetResponse and not only delta data related to the last position. Generally, the MDM supports no delta packages.

As an alternative to an inquireAll call, a client can also call the get method with the element value position = 0 to initialize itself or to collect at any time the latest available package.

For all three pull methods, the MDM ignores the following elements of the request from the OCIT-C protocol:

- username
- password
- watchdog

The filterList attribute in the call is not supported by any of the three methods and must always be requested empty from the data client system.

A client pull must always be uniquely assigned to a publication by referencing a subscription ID. This subscription ID, which is automatically assigned by the MDV of MDM, must be handed over by the data client system in the OCIT-C element <objectType>.

The following subsection is an example of a request for delivery in OCIT-C format for a fictitious subscription with the ID=2871015 of a fictitious organization „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>
```

The corresponding inquireAllResponse contains a data list with exactly one element of the DATEX II type D2LogicalModel. In this context, the MDM sets the following OCIT-C elements as follows:

- lastStart = an undefined, constant time the client should ignore
- errorCode = 0
- errorText = without content
- storetime/tstore = time of publication delivery at the MDM
- position = content ID of current data packet
- objectState = modified
- ident = none
- data = DATEX II payload

The following subsection shows a sample response. The DATEX II payload is shown in abbreviated form.

```
<?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>
        </ds>
      </dataList>
    </inquireAllResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

```

<data xsi:type="ns1:anyD2LogicalModel"
      xmlns:ns1="https://odg_und_partner/OCIT_C/Datex"
      xsi:schemaLocation=" https://odg_und_partner/OCIT_C/Datex
                           https://bast.s3.amazonaws.com/
                           schema/1446644360562/OcitCDatex2.xsd">
  <d2LogicalModel modelBaseVersion="2" extensionName="MDM"
                  extensionVersion="00-01-03"
                  xmlns="https://datex2.eu/schema/2/2_0"
                  xmlns:xsi="https://www.w3.org/2001/
                           XMLSchema-instance"
                  xsi:schemaLocation="
                           https://bast.s3.amazonaws.com/schema/1370439856400/
                           MDM-Profile_ParkingFacilityStatus.xsd">
    <exchange>
      <supplierIdentification>
        <country>de</country>
        <nationalIdentifier>DE-MDM-TEST</nationalIdentifier>
      </supplierIdentification>
    </exchange>
    <payloadPublication xsi:type="GenericPublication" lang="de"
                        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
      ...
    </payloadPublication>
  </d2LogicalModel>
</data>
</ds>
</dataList>
</inquireAllResponse>
</soapenv:Body>
</soapenv:Envelope>

```

With the help of the item <position> from the inquireAllResponse, the data client system can parameterize the get or wait4Get method to read subsequent packets.

A get call should always be distinctly assigned to a subscription by referencing a subscription ID and be assigned to a data packet by referencing the content ID. This subscription ID must be handed over by the data client system in the OCIT-C attribute <objectType> and the Content-ID in the attribute <position>. The MDM does not support a get call using a start and end time.

The following example shows a get request for delivery in OCIT-C format for a fictitious subscription (ID=2871015) and a fictitious preceding content ID=3876098:

```

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

```



The MDM then forms the `getResponse` and `wait4GetResponse` using the same attributes as in the `inquireAllResponse`.

For the `wait4Get` call, the same requirements apply as for the regular `get` call. In addition, the data client system must transmit the timeout value of the client in the `<MaxWaitTime>`. If this element is not submitted, the `wait4Get` request behaves like a regular `get` request and immediately sends an empty response, if no data package has been received in the meantime. If this value is above the 120 second maximum set in the MDM, the default MDM timeout is applied and the requester receives a response after a maximum of 120 seconds. This response is empty, if no data package has been received during the waiting interval.

The MDM does not support the option to read different objects by means of a single `wait4Get` request. Thus, only one subscription can be queried with a `wait4Get` call. List queries are rejected with an error.

The following example shows a `wait4Get` request for delivery in OCIT-C format for a fictitious subscription (ID=2871015), the fictitious content ID=3876098 and `maxWaitTime=60` seconds.

```
<?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>
```

This is an example of a `wait4Get` response from the MDM:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  <soapenv:Header/>
  <soapenv:Body>
    <ocit:wait4GetResponse xmlns:ocit="http://odg_und_partner/OCIT_C">
      <ocit:lastStart>2021-07-22T15:37:38.000+02:00</ocit:lastStart>
      <ocit:errorCode>0</ocit:errorCode>
      <ocit:errorText></ocit:errorText>
      <ocit:waitResponseList>
        <ocit:storetime>2021-08-02T11:53:51.480+02:00</ocit:storetime>
        <ocit:objectType>0</ocit:objectType>
        <ocit:position>0</ocit:position>
        <ocit:dataList>
          <ocit:ds>
            <ocit:tstore>2021-08-02T11:53:51.480+02:00</ocit:tstore>
            <ocit:objectState>modified</ocit:objectState>
            <ocit:identifier>
              <ocit:ident>None</ocit:ident>
            </ocit:identifier>
            <ocit:data xsi:type="ns1:anyD2LogicalModel"
              xmlns:ns1="http://odg_und_partner/OCIT_C/Datex"
              xsi:schemaLocation="http://odg_und_partner/OCIT_C/Datex
```

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

<http://bast.s3.amazonaws.com/schema/1446644360562/OcitCDatex2.xsd></>

The delivery of data packets at the MDM is generally GZIP compressed. This also applies to delivery via OCIT-C protocol. Data client systems must therefore decompress the packages in the web server before they can be processed using the OCIT-C protocol.

5.3.4 Error Handling

The following OCIT-C error codes are used:

Error code	Description
access error (1)	fundamentally incorrect parameterization of requests
internal error (22)	error in the internal processing of the request
missing parameters to execute the method (23)	missing subscription ID with get,wait4get or inquireAll

Table 9: OCIT-C error codes used

## 6 DATEX II v3

Contrary to DATEX II v2 Exchange, data transport in DATEX II v3 Exchange 2020 is realised by means of a MessageContainer structure. The MDM does not support all of the elements allowed by the DATEX II v3 specification, therefore the term “Minimal MessageContainer” is used. This container must include a payload element and an exchangeInformation element. The Minimal MessageContainer and the exchangeInformation element are available for download under [DATEXIIv3Exc].

The exchangeInformation element consists of two data structures with the following mandatory attributes:

- exchangeContext:
  - codedExchangeprotocol: An attribute of an enumeration type; its value depends on the applied protocol:
    - For SOAP interfaces, either „snapshotPull“ or „snapshotPush“ are used.
    - For HTTP Pull, „snapshotPull“ is used.
  - exchangeSpecificationVersion: The MDM expects the value „3.0“.
  - supplierOrCisRequester: To be standard-compliant, an empty XML element has to be included here.
- dynamicInformation:
  - exchangeStatus: Here, the fix value „online“ is generally expected.
  - messageGenerationTimestamp: Current timestamp of message generation.

### 6.1 XML Schema Application Notes for Exchange 2020

To create a DATEX II v3 publication, the data supplier must provide several XML schemas on two different levels:

1. **Data content:** DATEX II v3 introduces a namespace concept to DATEX II. With the creation of a publication data profile, a separate XML schema is generated for each namespace. Every instance of the publication must include a reference to the entry schema. This entry schema depends on the required compatibility level: DATEXII\_3\_D2Payload.xsd is the schema to be used for Level A or B publications while LevelC\_3\_D2Payload.xsd is reserved for Level C publications. Via these schemas, any additional schema of the according data profile is imported.
2. **Protocol data:** For transporting DATEX II v3 contents using the corresponding DATEX II Exchange 2020 specification, the MDM includes the schema of the content data in two additional schemas: MessageContainer.xsd and ExchangeInformation.xsd.

Both protocol data schemas have been profiled for MDM application. It is important to understand that the schema of the DATEX II MessageContainer object is where protocol data and the content data are combined. Therefore, this schema must be

adapted to handle Level A/B content or Level C content. How to proceed in both cases is described in the following.

#### 6.1.1 DATEX II v3 Level A or B

Several schemas are created for the XML schema of the publication data profile. Here, DATEXII\_3\_D2Payload.xsd is the schema to be included in each instance of the publication. It defines the namespace <http://datex2.eu/schema/3/d2Payload>. The data supplier must upload these content data schemas for his publication in the MDM graphical user interface together with the ExchangeInformation.xsd and MessageContainer.xsd variants for Level A und B publications [DATEXIIv3Exc].

#### 6.1.2 DATEX II v3 Level C

Several schemas are created for the XML schema of the publication data profile. Here, LevelC\_3\_D2Payload.xsd is the schema to be included in each instance of the publication. It defines the namespace <http://levelC/schema/3/d2Payload>. The data supplier must upload these content data schemas for his publication in the MDM graphical user interface together with the ExchangeInformation.xsd and MessageContainer.xsd variants for Level C publications [DATEXIIv3Exc].

i

### Important Note

In the MDM context, DATEX II v3 content has to be based on an XML entry element derived from the abstract class PayloadPublication in the DATEX II v3 package Common. As the Exchange 2020 MessageContainer expects this object, it does not matter here which compatibility level is applied.

The XML schema generated from the according DATEX II package Common always must be part of the content data schemas, i.e. either DATEXII\_3\_Common.xsd (Level A or B) or LevelC\_3\_Common.xsd (Level C).

Users who want to create a Level C publication with a different structure have to carry out modifications on XML schema level. For assistance, please contact the MDM Support.

## **6.2 SOAP Interface**

### **6.2.1 Data Supplier**

#### **6.2.1.1 Client Pull SOAP**

As with the Client Pull SOAP exchange process, the MDM broker system requests the data supplier system to deliver its data to the MDM platform.

##### **6.2.1.1.1 Offering a Web Service**

The data supplier system must provide a web service that is defined according to the DATEX II Snapshot Pull WSDL [DATEXIlv3Pull]. Null is thereby expected as input. As output, the MDM broker system expects the requested data inside a MessageContainer in DATEX II format according to the Minimal-MessageContainer profile in the MessageContainer.xsd schema [DATEXIlv3Exc].

It is the responsibility of the data supplier to define the mandatory exchangeInformation element with its exchangeContext and dynamicInformation elements. To provide the data in compliance with the standards, it is important to configure the value „snapshotPull“ for the ExchangeProtocol element. Irrespective of that, the MDM replaces the codedExchangeProtocol element by the value corresponding to the delivery protocol (see chapter 4.1) to provide correct data processing to standard conforming data client systems.

In the publication configuration page of MDM graphical user interface, the data supplier must configure the URL of his service endpoint where the MDM can retrieve the data packet.

#### 6.2.1.1.2 Calling up a Web Service

The MDM broker system provides a web service client that is defined according to the DATEX II Snapshot Pull WSDL [DATEXIIv3Pull] to invoke web services. This web service must return data according to the schema MessageContainer.xsd [DATEXIIv3Exc].

In case there is no data packet available for delivery in the data supplier system, the MDM broker system expects the following response:

- HTTP Response Code 200 Ok
- A Minimal MessageContainer element without payload

#### Response example:

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

The broker system identifies the data supplier systems that have subscribed to a pull method and the associated service endpoints in the metadata directory and periodically calls them up according to the configured publication frequency. The data received after the call is cached in corresponding packet buffers to be provided to potential data clients. A previous data packet, if it still exists, will be replaced.

### 6.2.1.2 Publisher Push SOAP

With the Publisher Push exchange process, the data supplier system must deliver the data to the MDM platform on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is irrelevant to the operation of the MDM broker system. The mechanism for the exchange is the same in both cases.

#### 6.2.1.2.1 Offering a Web Service

The MDM broker system provides a web service that is defined based on the specification DATEX II Snapshot Push WSDL [DATEXIIv3Push]. As input, the data to be supplied is expected in a MessageContainer instance in the body element of the SOAP envelope.

It is the responsibility of the data supplier to define the mandatory exchangeInformation element with its exchangeContext and dynamicInformation elements. To provide the data in compliance with the standards, it is important to configure the value „snapshotPull“ for the codedExchangeProtocol element. Irrespective of that, the MDM replaces the codedExchangeProtocol element by the value corresponding to the delivery protocol (see chapter 4.1) to provide correct data processing to standard conforming data client systems.

In the service endpoint URL of the broker system, the ID of the target publication for the data packets is entered.

The URL is structured as follows:

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

#### Example:

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

```

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

```

#### 6.2.1.2.2 Calling up the Web Service

The data supplier system has to provide a web service client that is defined according to DATEX II Snapshot Push WSDL [DATEXIIV3Push] to call up the web service. The web service must deliver the data to the publication-specific service endpoint of the MDM broker system. The MDM broker system accepts this data and stores it in a packet buffer. A previous data packet, if it still exists, will be replaced.

If the data transfer could be successfully completed, the MDM responds via DATEX II ExchangeInformation with the positive returnStatus “ack” according to the schema definition in ExchangeInformation.xsd [DATEXIIV3Exc].

#### Example:

```

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

```



If the data transfer could, however, not be successfully completed, the MDM responds via DATEX II ExchangeInformation with the negative returnStatus “fail” according to the schema definition in ExchangeInformation.xsd [DATEXIIv3Exc], e.g., if the publication is not configured for SOAP Push.

**Example:**

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

## 6.2.2 Data Client

### 6.2.2.1 Client Pull SOAP

With the Client Pull SOAP exchange process, the data client system must prompt the MDM platform to submit its data.

#### 6.2.2.1.1 Offering a Web Service

The MDM broker system provides a web service that is based on the specification [DATEXIIv3Pull]. As input, the subscription ID is expected in the URL. As output, the data client receives the requested data in a payload element of a MessageContainer in DATEX II Exchange 2020 format. Based on the transmitted subscription ID, the MDM platform can find the corresponding packet buffer and the data packet.

**Note:** If, at the time of request, the packet buffer does not include a data packet, the MDM responds via MessageContainer without a payload element (see also 6.2.1.1.2).

### 6.2.2.1.2 Calling up the Web Service

The data client system must provide a web service client that is defined according to the specification [DATEXIIv3Pull] to invoke web services. The corresponding subscription ID must be entered in the URL as input parameter.

The SOAP endpoint of the broker system is as follows:

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

#### Example:

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

### 6.2.2.2 Publisher Push SOAP

With the Publisher Push exchange process, the MDM broker system delivers the data to the data client systems on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is in this case irrelevant; the mechanism for the delivery to the data client is identical.

#### 6.2.2.2.1 Offering a Web Service

The data client system must provide a web service that is defined according to the specification [DATEXIIv3Push]. As input, the MDM sends the requested data via MessageContainer in the body element. The MDM platform expects a DATEX II ExchangeInformation as response with a positive returnStatus "ack" according to the schema definition in ExchangeInformation.xsd [DATEXIIv3Exc].

#### 6.2.2.2.2 Calling up the Web Service

The MDM broker system provides a web service client that is defined according to [DATEXIIv3Push] to invoke the web services of the data client system. Via the MDM administration component, the data client must enter its service endpoint in the subscription configuration.

The broker system identifies the data client systems and launches a corresponding web service call.

If the data transfer could be successfully completed, the broker system would then expect a confirmation message from the data client system:

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

## 6.3 HTTPS Interface

### 6.3.1 Data Supplier

#### 6.3.1.1 Client Pull HTTPS

As with the client pull exchange process, the MDM broker system periodically requests the data supplier system to deliver its data to the MDM platform. The time interval used must be configured in the metadata directory when configuring the data services. For this exchange, the Snapshot Pull rules from [DATEXIIv3Annex], *appendix C - "Snapshot Pull with simple http server" profile definition* apply.

It should be noted that the additional, optional rules do not apply. The options for authentication ([DATEXIIv3Annex], *appendix C - "Snapshot Pull with simple http server" profile definition, Authentication*) do not apply, as they are obsolete when using the HTTPS method that is compulsory for MDM. See also Appendix B – DATEX II HTTP Protocol Support.

##### 6.3.1.1.1 Request to Data Supplier

The MDM broker system sends an HTTPS GET request to the data supplier system from which the data is to be collected. The MDM platform is able to identify data supplier systems that have subscribed to a pull method, and to send requests to them at defined intervals.

Via the MDM administration component, the data provider must enter the publication-specific server URL in the publication configuration.

Also, consider chapter 4.5, Usage of the „If-Modified-Since“ Header Field.

##### 6.3.1.1.2 Response to the MDM Platform

After receipt of the request, the data supplier system must generate an HTTPS response whose message body consists of the requested DATEX II v3 data. Here, a MessageContainer object is expected, which complies to the the MessageContainer.xsd minimal profile [DATEXIIv3Exc]. Pursuant to [DATEXIIv3Annex], *appendix C - "Snapshot Pull with simple http server" profile definition, Basic request / response pattern*, the response must have the content type "text/xml; charset=utf-8" and can be submitted as GZIP encoding.

The MDM broker system accepts this data and stores it in a packet buffer. A previous data packet, if it still exists, will be replaced.

#### Example:

```
<?xml version='1.0'?>
<con:messageContainer xmlns:con='http://datex2.eu/schema/3/messageContainer'
  xmlns:ex='http://datex2.eu/schema/3/exchangeInformation'
  xmlns:d2='http://datex2.eu/schema/3/d2Payload'
  xmlns:loc='http://datex2.eu/schema/3/locationReferencing'
  xmlns:com='http://datex2.eu/schema/3/common'
  xmlns:sit='http://datex2.eu/schema/3/situation'
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xsi:schemaLocation='http://datex2.eu/schema/3/messageContainer
    ./DATEXII_3_MessageContainer.xsd'
  modelBaseVersion='3'>
  <con:payload lang='en'
    xsi:type='sit:SituationPublication'
```

```

        modelBaseVersion='3'>
    ...
</con:payload>
<con:exchangeInformation modelBaseVersion='3'>
    <ex:exchangeContext>
        <ex:codedExchangeProtocol>snapshotPull</ex:codedExchangeProtocol>
        <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
        <ex:supplierOrCisRequester/>
    </ex:exchangeContext>
    <ex:dynamicInformation>
        <ex:exchangeStatus>online</ex:exchangeStatus>
        <ex:messageGenerationTimestamp>2021-07-21T13:00:00
        </ex:messageGenerationTimestamp>
    </ex:dynamicInformation>
</con:exchangeInformation>
</con:messageContainer>

```

### 6.3.2 Data Client

#### 6.3.2.1 Client Pull HTTPS

With the client pull exchange process, the data client system must prompt the MDM broker system to transfer the data.

##### 6.3.2.1.1 Request to the MDM Platform

The data client system shall send an HTTPS GET request to the URL of the MDM platform. Due to the subscription ID, the associated packet buffer and the data packet are determined. Alternatively, an HTTPS POST request can be used.

The URL of the broker system is constructed as follows:

```

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

```

Also, consider chapter 4.5, Usage of the „If-Modified-Since“ Header Field.

### 6.3.2.1.2 Response to Data Client

The MDM broker system generates an HTTPS response after receipt of the request. For this purpose, the associated packet buffer and the appropriate data package will be determined based on the subscription ID. The content of the data packet is sent to the data client in the body of the response. Pursuant to DATEX II Client Snapshot Pull profile ([DATEXIIv2PSM], appendix C – “*Snapshot Pull with simple http server*” profile definition, Overall presentation), the content is always delivered with a MessageContainer instance. The response also has the content type „text/xml; charset=utf-8“ and is always sent GZIP compressed, deviating from the standard. **Requestes using “identity encoding” or other compression formats will be acknowledged with an HTTP 406 (Not Acceptable) error code.**

Default HTTP status codes may occur according to [HTTP/1.1], while their significance is given in Table 10:

Description	
Request	Request GET /BASt-MDM- Interface/dtexv3/http/content.xml?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx < messageContainer > ... </messageContainer>
Status codes	Standard HTTP1.1 status codes [HTTP/1.1] The following status codes have special significance: - 304: not modified, the requested resource is not transmitted again - 401: authentication error - 204: no data packet in subscription packet buffer - 404: no subscription or invalid subscription found for subscription parameter - 406: ressource has not been requested in GZIP format - 503: service unavailable (e.g., in case of maintenance)

Table 10: Request/Response between MDM platform and data client system (Client Pull HTTPS)

## **7 Container**

### **7.1 SOAP Interface**

#### **7.1.1 Data Supplier**

##### **7.1.1.1 Client Pull SOAP**

As with the client pull SOAP exchange process, the MDM broker system periodically requests the data supplier system to deliver its data to the MDM platform. The time interval used must be configured in the metadata directory when configuring the data services.

##### **7.1.1.1.1 Offering a Web Service**

The data supplier system has to offer a web service with the method `putContainerDataBroke` that expects as input the parameters publication ID (type `publicationId`) and an optional time stamp with the date of creation according to the elements of the container model schema. The data supplier system must generate and return a data packet (type `containerdata`) in the container format for the transferred publication ID.

Via the MDM administration component, the data supplier must enter the service endpoint in the URL attribute of the publication configuration.

##### **7.1.1.1.2 Calling up a Web Service**

The MDM broker system provides a web service client that is defined according to the container format specification [MCS] to invoke web services.

The broker system identifies the data supplier systems that have subscribed to a pull method and the associated service endpoints in the metadata directory and periodically calls them up according to the configured publication frequency. The data received after the call is cached in corresponding packet buffers for delivery to potential data clients. A previous data packet, if it still exists, will be replaced.

### 7.1.1.2 Publisher Push SOAP

With the Publisher Push exchange process, the data supplier system must deliver the data to the MDM platform on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is irrelevant to the operation of the MDM broker system. The mechanism for the exchange is the same in both cases.

#### 7.1.1.2.1 Offering a Web Service

The MDM broker system provides a web service with the method `pushContainerData` which expects - as input - the data structure of the container format filled with the publication ID in the header element and a data packet in the body element and returns a status message as output. An object of the type `containerdata` is expected in each case.

#### Example:

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

#### 7.1.1.2.2 Calling up the Web Service

The data supplier system must provide a web service client in accordance with the container format specification [MCS]. This client serves to launch the web service.

The SOAP endpoint of the broker system is as follows:

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



## 7.1.2 Data Client

### 7.1.2.1 Client Pull SOAP

With the Client Pull SOAP exchange process, the data client system must prompt the MDM platform to transfer the data to the data client system.

#### 7.1.2.1.1 Offering a Web Service

The MDM broker system provides a web service with the method `pullContainerDataClient`, which expects - as input - a subscription ID (type `subscriptionId`) in the XML data and an optional timestamp (type `timestamp` – it includes the creation time of the request). As output, the data is returned in container format (type `containerdata`).

**Example:**

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

#### 7.1.2.1.2 Calling up the Web Service

The data client system must provide a web service client in accordance with the container format specification [MCS]. This client serves to launch the web service.

The SOAP endpoint of the broker system is as follows:

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

### 7.1.2.2 Publisher Push SOAP

With the Publisher Push exchange process, the MDM broker system delivers the data to the data client systems on its own initiative. In this process, an appropriate SOAP interface must be used. Whether the data is event-based (on occurrence) or periodically generated and delivered to the MDM platform is in this case irrelevant; the mechanism for the delivery to the data client remains identical.

#### 7.1.2.2.1 Offering a Web Service

The data client system must provide a web service with the method `pushContainerData` that is defined on the basis of the container format specification [MCS]. A data packet of the type container format (type `containerdata`) must be accepted as input and, as output, a status message (also of type `containerdata`) for delivery.

#### **7.1.2.2.2 Calling up the Web Service**

The MDM broker system provides a web service client that is defined according to the container format specification [MCS] to invoke the web services of the data client system. Via the MDM administration component, the data client must enter its service endpoint in the URL attribute of the subscription configuration.

The broker system identifies the data client systems and launches a corresponding web service call.

If the data transfer could be successfully completed, the broker system would then expect a status message from the data client system.

## 7.2 HTTPS Interface

### 7.2.1 Data Supplier

#### 7.2.1.1 Client Pull HTTPS

The MDM broker system prompts the data supplier system to periodically deliver a packet for a publication to the MDM platform. The time interval used must be configured in the metadata directory when configuring the data services.

##### 7.2.1.1.1 Request to Data Supplier

The broker system sends an HTTPS GET request to the data supplier system. As a parameter, the publication ID of the publication for which a data packet has to be delivered is handed over to the MDM. Via the MDM administration component, the data supplier must enter its URL in the publication configuration.

The URL of the data supplier system from the publication configuration is complemented by appending the publication ID.

##### Example:

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

##### 7.2.1.1.2 Response to the MDM Platform

The data supplier system must respond to the request with an HTTPS response. The content type of the response must be of the type "text/xml" and should be available as GZIP encoding. Non-compressed content can also be processed by the MDM platform. The message body has to include the requested data packet. The standard HTTP status codes [HTTP/1.1] must be used, whereby the explanations described in Table 11 shall apply.

Description	
Request	GET /anfrageServlet?publicationID=2000002 HTTP/1.1 Host: Datengeberhost Accept-Encoding: GZIP
Response	HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <container> ... </container>
Status codes	Standard HTTP1.1 Statuscodes [HTTP/1.1] The following status codes have special significance: - 400: no publication parameter has been submitted - 404: publication parameters could not be assigned

Table 11: Request/Response between data supplier system and MDM platform with client pull HTTPS

### Example:

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

#### 7.2.1.2 Publisher Push HTTPS

The data supplier system has to send a data packet for a publication to the MDM broker system.

##### 7.2.1.2.1 Request to the MDM broker system

The data supplier system must send an HTTPS POST request with a message in container format to the MDM broker system. In this process, the publication ID must be delivered in the header element and the payload must be delivered in the body element of the container message.

The URL of the broker system is constructed as follows:

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

### Example:

```
<?xml version='1.0' encoding='UTF-8'?>
<ns3:containerRootElementEl xmlns="https://www.w3.org/2000/09/xmldsig#"
  xmlns:ns2="https://schemas.xmlsoap.org/ws/2002/07/utility"
  xmlns:ns3="https://ws.bast.de/container/TrafficDataService">
  <ns3:header>
    <ns3:Identifier>
      <ns3:publicationId>12345</ns3:publicationId>
    </ns3:Identifier>
  </ns3:header>
  <ns3:body>
    <ns3:binary id="test-id-bin" type="hexBinary">
      dGVzdC10ZXh0&#xD;.
    </ns3:binary>
    <ns3:xml schema="test-schema" id="test-id-xml">
      <n4:musterDatenRoot>
        <n4:trafficData origin="home"/>
      </n4:musterDatenRoot>
    </ns3:xml>
  </ns3:body>
</ns3:containerRootElementEl>
```

#### 7.2.1.2.2 Response an den Datengeber

To the request, the data supplier system receives an HTTPS response. The message body is empty. The standard HTTP status codes [HTTP/1.1] may occur as status codes, where the explanations given in Table 12 shall apply.

Description	
Request	Request POST /datenabgabe HTTP/1.1 Host: mdmhost Content-Type : text/xml Accept-Encoding: GZIP <container> ... </container>
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 Statuscodes [HTTP/1.1] The following status codes have a special significance: - 400: no publication parameter or no data submitted - 404: publication parameters could not be assigned or publication no longer valid

Table 12: Request/Response between data supplier system and MDM platform with publisher push HTTPS

### 7.2.2 Data Client

#### 7.2.2.1 Client Pull HTTPS

With the client pull exchange process, the data client system must prompt the MDM broker system to transfer the data. The according subscription has to be specified by a request parameter.

##### 7.2.2.1.1 Request to the MDM Platform

The data client system must send an HTTPS GET request to the MDM platform. As a parameter, the ID of the subscription for which a data packet has to be delivered must be submitted to the MDM.

The URL of the broker system is constructed as follows:

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

##### 7.2.2.1.2 Response an das Datennehmersystem

The MDM broker system generates an HTTPS response after receipt of the request. The standard HTTP status codes [HTTP/1.1] can be used, where the explanations described in Table 13 shall apply. The content type of the response is of the type "text/xml" and is sent GZIP-compressed. The message body of the response consists of the requested data packet.

Description	
Request	Request GET /BASt-MDM-Interface/srv/container/v1.0?subscriptionID=2000000 HTTP/1.1 Host: mdmhost Accept-Encoding: GZIP
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <container> ... </container>
Status codes	Standard HTTP1.1 Statuscodes [HTTP/1.1] The following status codes have a special significance: - 204: no data packet in subscription packet buffer - 400: no subscription parameter - 404: no subscription or invalid subscription found for subscription parameter

Table 13: Response between MDM platform/data client system with Client Pull HTTPS

## 7.2.2.2 Publisher Push HTTPS

The MDM broker system sends a data packet of a subscription to a data client system.

### 7.2.2.2.1 Request to the Data Client System

The MDM broker system sends an HTTPS POST request to the data client system in which the subscription ID is submitted in the header element and the user data is submitted in the body element of the container message.

Via the MDM administration component, the data client must enter its URL in the subscription configuration.

### 7.2.2.2.2 Response to the MDM Platform

The data client system must respond to the request with an HTTPS response.

The message body is empty. The standard HTTP status codes [HTTP/1.1] may be used as status codes, where the explanations described in Table 14 shall apply.

Description	
Request	Request POST /data delivery HTTP/1.1 Host: data client host Content-Type : text/xml

	Accept-Encoding: GZIP <container> ... </container>
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 status codes [HTTP/1.1] The following status codes have special significance: - 400: no subscription parameter or no data submitted - 404: subscription parameter could not be assigned

*Table 14: Request/Response between MDM broker system/data client system with Publisher Push HTTPS*

## 8 Certificate-based M2M Communication

The security component of the MDM platform requires a certificate-based data exchange between the data supplier system and the platform, on the one side, and between the platform and the data client system, on the other.

This chapter begins with an overview of the functions of the security component. Then it describes the steps to be taken by the data providers and the data clients to request certificates and set them up for M2M communication.

The certificate is created following a request and then sent to the data supplier/data client by e-mail. The password that is required for signature is sent via SMS.

The data supplier system/data client system must finally integrate the certificate into their IT infrastructure, so that the data exchange with the MDM platform can be authenticated.

### 8.1 Tasks of the Security Component

The security component is responsible for the realization of the safety aspects of the MDM platform. This includes, in particular, the authentication of data supplier systems and data client systems, which want to communicate with the MDM platform.

Before the data packets arriving at the MDM platform can be accepted, their origin must be checked. This includes the authentication of the data supplier system that is associated with the data packet using a digital certificate. Each data supplier system must have a valid certificate to be used for login at the platform. The security component authenticates the certificate sent by the data supplier system within the MDM platform.

Before a data packet can be sent to a data client system, the identity of this data client system needs to be checked. Each data client system must authenticate itself to the MDM platform using a digital certificate. The security component authenticates the certificate sent by the data client system within the MDM platform.

The confidentiality of communication between the data supplier system and the MDM platform, on the one hand, and the MDM platform and the data client system, on the other hand, must be ensured by an exclusive use of an SSL/TLS transport encryption.

The security component requires standards-compliant [X.509v3] certificates for authentication; see also [PKI]. The certificates must be technically involved in the HTTPS connection to the data client and data supplier systems via a client-side, certificate-based connection establishment. The presented certificates are checked for validity and whether they are blocked or not.



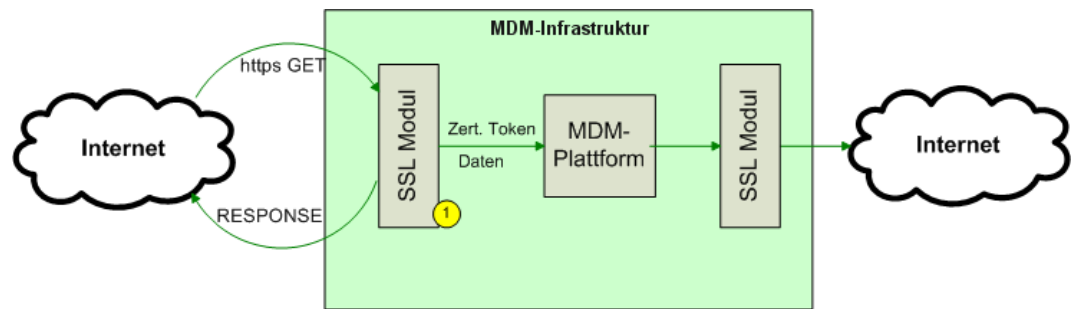


Figure 4: Overview of the security architecture

The SSL module 1 in Figure 4 sends a certificate request to the sender for predefined URLs, checks the validity of the obtained certificate and then verifies whether it is blocked or not. Afterwards, it forwards the certificate to the security component of the MDM platform.

## 8.2 Note on Server Name Indication

The MDM platform does not support the Server Name Indication (SNI) feature.

This means that data suppliers for the client pull method and data clients for the publisher push method cannot use any virtual server for M2M communication. Each registered machine can represent only a unique IP address.

## 8.3 Applying for a Machine Certificate

The operator of the MDM platform mediates between the data supplier or data client systems and the certificate issuer. Therefore, data providers and data clients apply - when registering - for one or multiple machine certificates via the administration GUI of the MDM platform. The certificate is however sent to them by the certificate-issuing organization and not by the operator of the MDM platform.

To request a machine certificate, you must already be registered on the MDM platform with your organization.

How to apply for a machine certificate on the MDM platform is described in [BHB].

## 8.4 Installing a Machine Certificate and Issuer Certificate

In the Apache Web server, integrate the machine certificate as follows:

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

Enter the associated private key as follows:

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

In addition, you must define the issuer certificate on the web server:

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

The certificate is encrypted by using the key with the password that has been sent to you via fax. Use the password to decrypt.

For more information on these directives, please see the mod\_ssl documentation:

[http://httpd.apache.org/docs/current/mod/mod\\_ssl.html#sslcertificatefile](http://httpd.apache.org/docs/current/mod/mod_ssl.html#sslcertificatefile)

[http://httpd.apache.org/docs/current/mod/mod\\_ssl.html#sslcacertificatefile](http://httpd.apache.org/docs/current/mod/mod_ssl.html#sslcacertificatefile)

**Note:** If you get the machine certificate and the issuer certificate within a common p2 file, you must extract both certificates from this file and then install them. The relevant instructions are provided in chapter 9.

## 8.5 Authentication of the MDM Platform as Web Client

If the MDM platform acts as a web client in the M2M communication, it will then authenticate with its server certificate, provided that the web server has enabled this option on the data supplier or data client side. Data supplier and data client systems should enable this option and verify the certificate to determine that the requests were actually sent by the MDM platform.

The CA certificates required for verification can be downloaded from <https://service.mdm-portal.de/doc/MDM-CA-Bundle.zip> and must be stored in the data supplier or data client systems.

**Note:** Do not use the MDM server certificate for verification as it is changed on a regular basis.

## 8.6 Authentication of Data Supplier/Data Client Web Clients

If the data supplier or data client systems act as a web client in the M2M communication, the web client must authenticate to the MDM platform by using its machine certificate. The platform will accept requests only from systems that are registered in the metadata directory. Based on the certificate, the machine can be associated with the organization. Furthermore, it can be checked whether the organization is the owner of the publication or subscription for which data exchange is to take place.

The server certificate for broker.mdm-portal.de has been created by Comodo. In most cases, it will not be necessary to install the Comodo CA certificate when using the operating system's trust store. Nevertheless, it may be necessary to install the CA certificates of the MDM CA. An archive with all required certificates can be found at:

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

## 9 Appendix A- Processing the p12 File for Apache Server Configuration

The Apache server configuration cannot handle any files of the type p12. For processing, manual steps that are described in the following chapters are required:

First, export the keys and certificates. Run the following command from the command prompt:

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

### Example:

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

Enter the certificate passwords in the openssl environment:

```
>Enter Import Password:      <Password from fax>  
>MAC verified OK  
>Enter PEM pass phrase:      <Self-selected passphrase for the key>  
>Verifying - Enter PEM passphrase: <Repetition of the self-selected  
passphrase for the key>
```

Open the file <sammeldatei.pem> with a text editor:

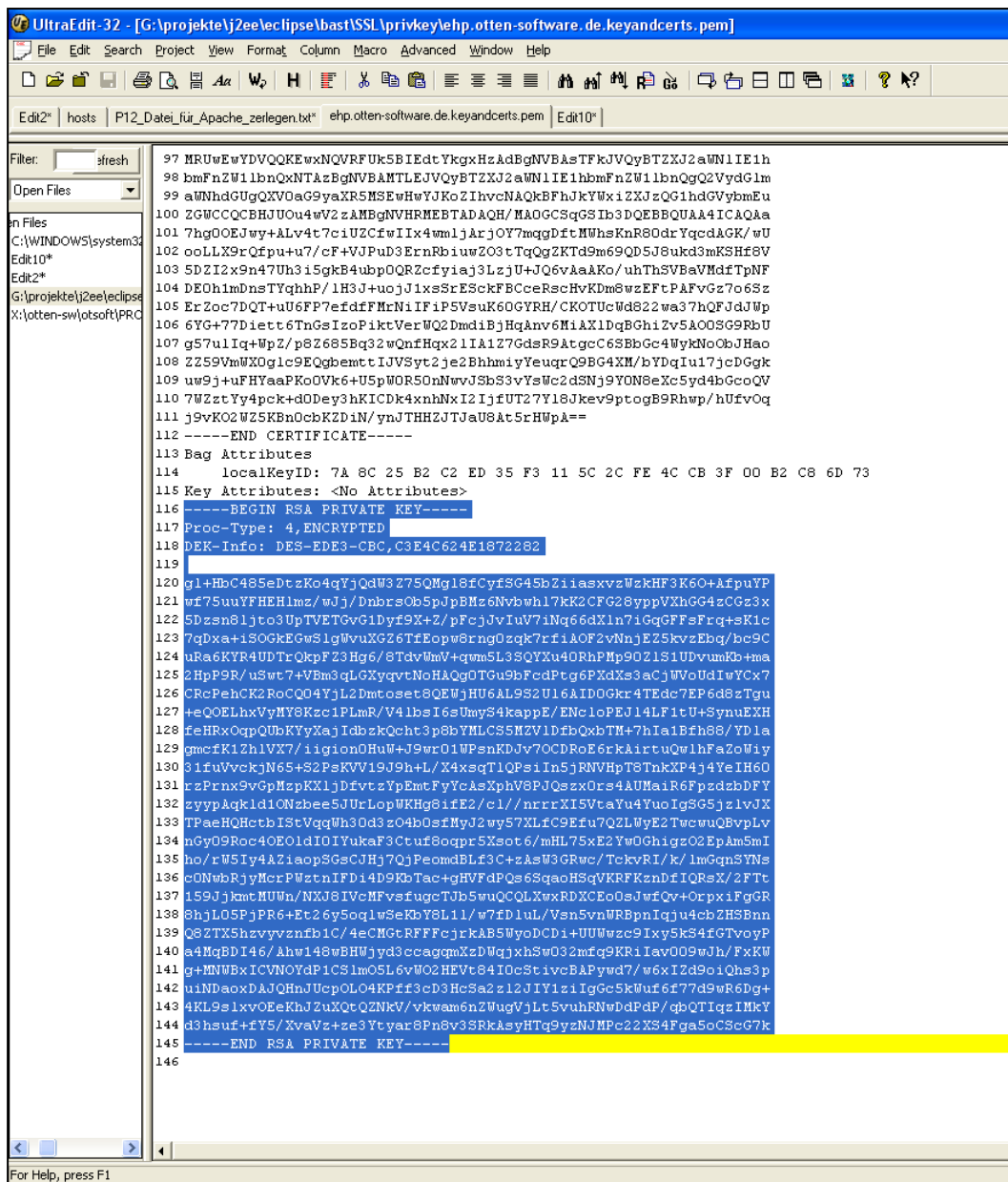


Figure 5: Datei <sammeldatei.pem>

Copy the part of

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

until

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

to a new file named <server.key>

Remove the passphrase to prevent that it is requested each time the server is restarted:

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

### Example:

```
openssl rsa -in server.key -out ehp.otten-software.de.key  
> Enter passphrase for server.key:<Enter the previously self-selected  
passphrase>  
>writing RSA key
```

Enter the generated .key file in the Apache configuration under the following attribute:

```
SSLCertificateKeyFile
```

Next, split the certificates into two files. To do this, first open the file <sammeldatei.pem> with a text editor:

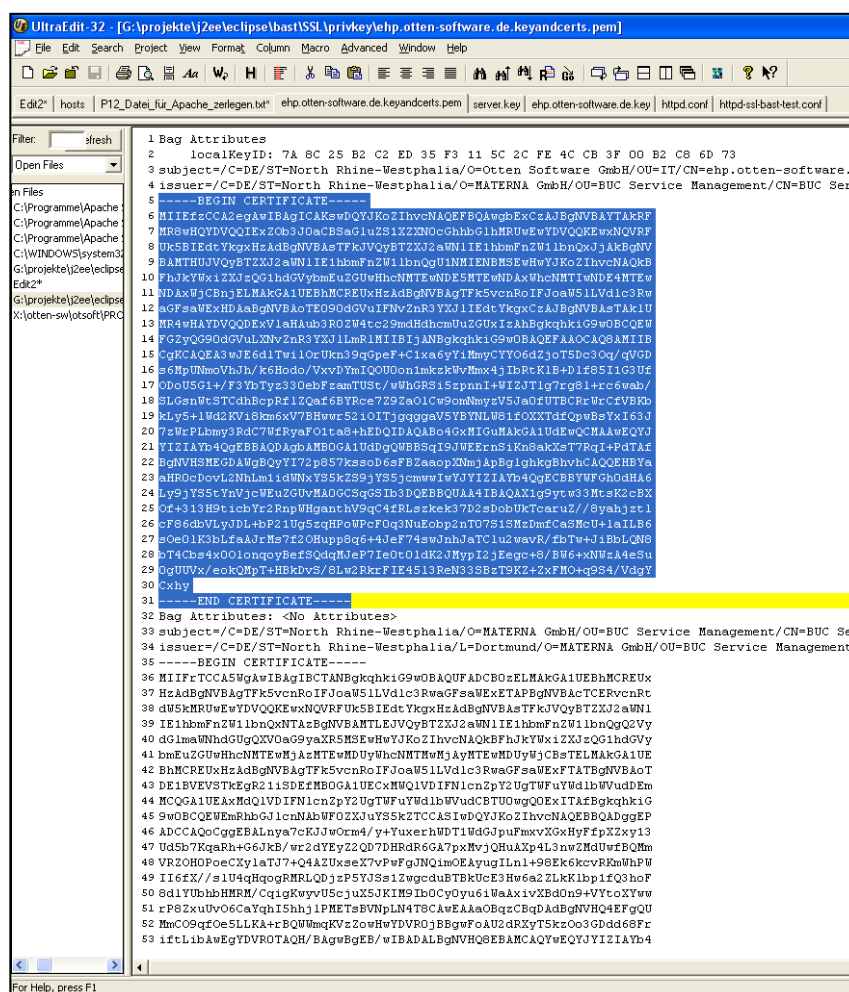


Figure 6: File <sammeldatei.pem>

Copy the server certificate into a new text file <server.crt>.

Enter this file in the Apache configuration under the following attribute:

```
SSLCertificateFile
```

Copy the remaining certificates into a new text file <ca-cert-chain.crt>.

Enter this file in the Apache configuration under the following attribute:

```
SSLCertificateChainFile
```

Enter the MDM client certificate incl. the certificate hierarchy under the following Apache attribute:

```
SSLCACertificateFile
```

Example of an Apache configuration:

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

## 10 Appendix B – DATEX II HTTP Protocol Support

Rule	Reference to rule in [DATEXIIv2PSM], chapter 4	Reference to rule 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.	✓	✓
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	C.7	Basic request / response pattern: 7.	✓	✓

Rule	Reference to rule in [DATEXIIv2PSM], chapter 4	Reference to rule in [DATEXIIv3Annex]	DATEX II v2	DATEX II v3
containing payload (response code 200) message into this field.				
Suppliers SHOULD provide XML coded DATEX II payload as “text/xml” media type. Suppliers SHOULD state the used character set via the “charset” parameter; Suppliers SHOULD use the UTF-8 character set, i.e., the “Content-Type” response-header field SHOULD state “text/xml; charset=utf-8.”	C.8	Basic request / response pattern: 8.	✓	✓
Clients MUST accept “identity” content-coding; Clients SHOULD (and if they do, prefer to) accept “gzip” content-coding; Clients MAY accept other “content-coding” values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry <sup>1</sup> as long as they also accept “identity” and “gzip” content-coding.	C.9	Basic request / response pattern: 9.	✓	✓
When including an “Accept-Encoding” request-header field in an HTTP REQUEST message, the Client MUST NOT exclude acceptance of “identity” content-coding.	C.10	Basic request / response pattern: 10.	✓	✓
Suppliers MUST provide “identity” content-coding of the payload; Suppliers SHOULD provide “gzip” content-coding of the payload; Suppliers MAY provide other “content-coding” values registered by the Internet Assigned Numbers Authority (IANA) in their content-coding registry as long as they also provide “identity” and “gzip” content-coding	C.11	Basic request / response pattern: 11.	✗	✗
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	✗	✗



Rule	Reference to rule in [DATEXIIv2PSM], chapter 4	Reference to rule 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		✗	
<p>Servers SHALL produce and Clients SHALL process the following return codes:</p> <ul style="list-style-type: none"> <li>• 200 (OK), in responses carrying payload,</li> <li>• 304 (Not Modified), if no payload is send because of the specification in the 'If-Modified-Since' header,</li> <li>• 503 (Service Unavailable), if an active HTTP server is disconnected from the content feed,</li> <li>• 404 (Not Found), if a file based HTTP server does not have a proper payload document stored in the place associated to the URL.</li> </ul>	C.15	Additional Rules	(✓) difference: 403 instead of 401	✓ additionally 204, in case of empty packet buffer
<p>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.</p>	C.16	Describing payload and interfaces	✗	✓