mobilithek

Germany's data platform
that gets you moving.

Technical interface description

Version 1.2.2 - 04.04.2024

Federal Ministry
for Digital
and Transport

Table of contents

List of tables

List of figures

1 Introduction

1.1 Abstract

The Mobilithek aims to support the exchange of data between data providers and data users with the help of interfaces and at the same time represents a central portal with the collected information on available online traffic data of individual data providers. In this way, the Mobilithek enables its users to offer, find and subscribe to traffic-relevant online data without the need for a lengthy search for the relevant data and time-consuming technical and organisational bilateral coordination between data users and data providers. The data exchange is handled via standardized interfaces. As a result, business processes are to be simplified for all parties involved and the potential of existing data sources is to be tapped.

This interface description is intended for potential data providers and data recipients. Knowledge in the implementation and operation of SOAP web services or [SOAP] or HTTPS client/server architectures are required to use the interfaces of the Mobilithek.

Data transmission between the Mobilithek and the data provider or data recipient systems can be carried out either via SOAP-based web services or simple HTTPS GET/POST requests. In addition, transmission via OCIT-C protocol is offered.

1.2 Structure of the document

The document is divided into the following chapters:

- Chapter 1 contains a brief overview, the referenced documents and the list of abbreviations.
- In the chapter 2 the components of the Mobilithek are presented.
- Chapter 3 deals with the available data formats.
- The interfaces of the Mobilithek for M2M communication are described in chapter 4 chapter.
- Chapter 5 describes interfaces that are independent of the data format.
- Chapter 6 describes the DATEX II v2 format in detail.
- Chapter 7 describes DATEX II v3 accordingly.
- Chapter 8 describes the container format of the Mobilithek.
- Chapter 9 describes the handling of other data formats.
- Chapter 10 describes the measures used to secure M2M communication.

1.3 Referenced documents

1.3.1 General

[Source]	Publisher
[FAQ]	"Frequently asked questions" https://mobilithek.info/help/FAQ
[GZIP]	RFC 1952 (May 1996) GZIP File Format Specification Version 4.3, https://tools.ietf.org/rfc/rfc1952.txt
[HTTP/1.1]	RFC 2616 (June 1999) Hypertext Transfer Protocol -- HTTP/1.1 https://www.ietf.org/rfc/rfc2616.txt
[HTTPS]	RFC 2818 (May 2000) HTTP over TLS https://www.ietf.org/rfc/rfc2818.txt
[MCS]	Container format specification In the download area (https://mobilithek.info/help/download) in the category Documentation; see Container Specification V1.1 - 02|2014
[OCIT-C]	OCIT-C Specification Version 1.1_R1 from 30.10.2014 https://www.ocit.org/media/ocit-c_protokoll_v1.1_r1.pdf In the download area (https://mobilithek.info/help/download) in the category WSDL; see Mobilithek: OCIT-C WSDL

[Source]	Publisher
[PKI]	RFC 2459 (January 1999) Internet X.509 Public Key Infrastructure Certificate and CRL Profile https://www.ietf.org/rfc/rfc2459.txt
[SOAP]	SOAP Version 1.2 https://www.w3.org/TR/soap12-part1/
[URL]	RFC 1738 (December 1994) Uniform Resource Locators (URL) https://www.ietf.org/rfc/rfc1738.txt
[X.509v3]	ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997. https://www.itu.int/rec/T-REC-X.509-199708-S/en

Table 1: Referenced documents (comprehensive)

1.3.2 DATEX II v2

[Source]	Publisher
[DATEXIIv2PSM]	DATEX II v2.0 Exchange Platform Specific Model https://docs.datex2.eu/_static/data/v2.0/DATEXII_-_ExchangePSM_0.pdf
[DATEXIIv2Pull]	DATEX II v2.0 Pull wsdl In the download area (https://mobilithek.info/help/download) in the category WSDL; see Mobilithek - Pull: DATEXII v2
[DATEXIIv2Push]	DATEX II v2.0 Push wsdl In the download area (https://mobilithek.info/help/download) in the category WSDL; see Mobilithek - Push: DATEXII v2
[DATEXIIv2Schema]	DATEX II XML Schema 2.3 https://docs.datex2.eu/_static/data/v2.3/DATEXIISchema_2_2_3_0.zip
[DATEXIIv2SDG]	DATEX II v2.3 Software Developers Guide: https://docs.datex2.eu/_static/data/v2.3/DATEXII-DevGuide.pdf
[DATEXIIv2Spec]	Includes the following documents, which are available for download on https://www.datex2.eu for all registered users: [DATEXIIv2PSM], [DATEXIIv2UG]
[DATEXIIv2UG]	DATEX II v2.0 User Guide: https://docs.datex2.eu/_static/data/v2.3/DATEXII-UserGuide.pdf

Table 2: Referenced documents (DATEX II v2)

1.3.3 DATEX II v3

[Source]	Publisher
[DATEXIIv3Annex]	Annexes to Platform Specific Model: https://docs.datex2.eu/exchange/2020/psm/annexes.html
[DATEXIIv3Pull]	DATEX II v3.0 Snapshot Pull wsdl In the download area (https://mobilithek.info/help/download) in the category WSDL; see Mobilithek - Pull: DATEXII v3
[DATEXIIv3Push]	DATEX II v3.0 Snapshot Push wsdl In the download area (https://mobilithek.info/help/download) in the category WSDL; see Mobilithek - Push: DATEXII v3
[DATEXIIv3Exc]	Download for Level A and Level B publications: In the download area (https://mobilithek.info/help/download) in the category DATEXII V3 Download for Level C publications: In the download area (https://mobilithek.info/help/download) in the category DATEXII V3
[DATEXIIv3ExcUG]	Exchange 2020 User Guide: https://docs.datex2.eu/exchange/2020/userguide/
[DATEXIIv3Spec]	Includes the following documents, which are available for download on https://www.datex2.eu or on the Mobilithek: [DATEXIIv3ExcUG], [DATEXIIv3Annex], [DATEXIIv3Pull], [DATEXIIv3Push], [DATEXIIv3Exc]

Table 3: Referenced documents (DATEX II v3)

1.4 List of abbreviations

Abbreviation	Resolution
BASt	Federal Highway Research Institute
GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
M2M	Machine-to-Machine
MDM	Mobility data marketplace
MDV	Metadata directory
OCIT	Open Communication Interface for Road Traffic Control Systems
PKI	Public Key Infrastructure

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

Table 4: List of abbreviations

2 Components of the Mobilithek at a glance

The Mobilithek is made up of four components, each of which performs different tasks.

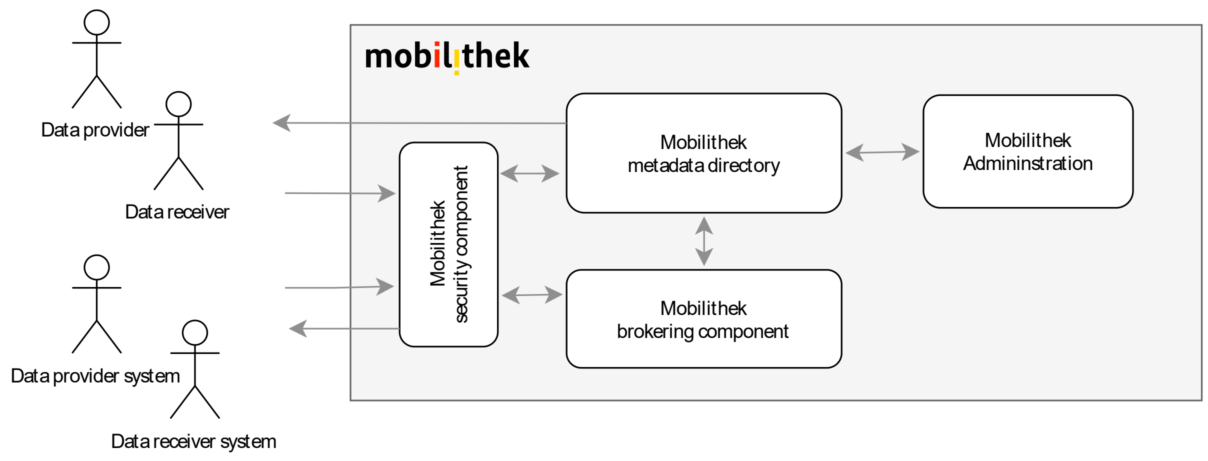

mobilithek
Data provider
Data receiver
Data provider system
Data receiver system
Mobilithek security component
Mobilithek metadata directory
Mobilithek Admininstration
Mobilithek brokering component

Figure 1: Components of the Mobilithek

Component	Description
Security component	The security component can be used to authenticate the data recipient system/data provider system or data recipient/data provider in order to be able to use services.
Metadata directory	The metadata directory is used to manage all information regarding the publications provided in the Mobilithek.
Broker system	The broker system handles the actual processing of the data packets and is therefore the focus of this interface description.
Administration	The administration is realised by means of a web-based user interface (GUI), see [FAQ].

Table 5: Overview of the components of the Mobilithek

The Mobilithek supports the following communication and application scenarios:

- Interested parties, data recipients and data providers can communicate with the metadata directory via the web GUI in order to make use of services such as searching or registering. In order to be able to view or change certain contents of the metadata directory, authentication must first be carried out at the Mobilithek Security component.
- After authentication via the security component, the data receiver system and data provider system can establish M2M communication with the broker system in order to deliver or request data.

3 Data exchange formats

In order to be able to exchange mobility data between broker system and data provider and data recipient system, the following data formats are specified:

- The Mobilithek supports the XML- or JSON-based DATEX II format through native interfaces to enable the platform to be used by standards-compliant DATEX II implementations at data providers or data recipients.
- In order to achieve the greatest possible compatibility with the MDM, the container format introduced with the MDM and independent of concrete formats continues to be supported. Any XML and binary data can be transmitted via this format.
- Furthermore, the HTTPS interface can in principle be used to transmit any "payloads" in the HTTP body without the need to package them in the container format beforehand.

When a data package is delivered to the broker interface of the Mobilithek, the validity of the data is checked for conformity with a stored file schema. A check for malware is also performed. The Mobilithek system administrator can disable these checks for individual publications.

The Mobilithek web GUI allows you to download the data package currently stored in the Mobilithek via a button.

If a validation of the data package is carried out, the file data is obtained via the URL stored in the publication description.

For publications in DATEX II format, it is the responsibility of the data provider to deposit the correct file schemas. For publications in container format, the standard schema is already made available under a generally valid URL. For data packages in XML format, the schema is to be referenced via the URL in the "schemaLocation" attribute to enable data recipients to automatically validate the packages with the same prerequisites. For data packages in JSON format, the schema is referenced in the $schema attribute. If this includes additional subschemas, these must be specified via the $ref attribute. The referenced subschemas must always referenced with their full name inclusive their file type suffix: *"$ref": "<subschema filename>"*. It is recommended to data providers to reference the schemas which are stored for the publication in Mobilithek, and to use those schemas for their own validations. The schema URLs can be obtained from the Mobilithek GUI in the publication detail screen. The benefit following that approach is that all parties of the data exchange (data provider, Mobilithek and data consumer) are using the same schemas for validation.

The Mobilithek accepts a data package regardless of its validity or the result of the check for malware and also delivers it to the data recipients if it is not valid in relation to the stored schemas or if there is a suspicion that it contains malware.

3.1 DATEX II

DATEX II is a pan-European standard for the exchange of mobility data. Basic knowledge of the DATEX II specification is assumed for this chapter [DATEXIIv2Spec] resp. [DATEXIIv3Spec]. The Mobilithek supports both DATEX II v2 and v3.

DATEX II defines XML structures for the exchange of mobility data. For DATEX II V3, JSON is supported as an exchange format in addition to XML. The underlying schema files can be obtained from the DATEX II website https://www.datex2.eu/. The user data are to be defined based on this schema. DATEX II not only specifies a standard for the structure of the user data, but also regulates the exchange process; this is described in more detail in chapter 4 is described in more detail.

The documents on which DATEX II is based are listed in chapter 1.3 "Referenced documents" as [DATEXIIv2Spec] resp. [DATEXIIv3Spec]. The structure of the DATEX II user data is not relevant for the mobile library, as it forwards the data unchanged and does not evaluate them.

DATEX II not only provides for sending complete data packages, but also for sending changes to previous versions. The Mobilithek basically supports the processing of "delta" submissions (see chapter 4.3). With regard to DATEX II, this option is currently only supported for v3 using XML. For DATEX II v2 and DATEX II v3 (JSON), the data provider system must always provide complete data packets that are forwarded unchanged by the Mobilithek.

3.2 Container format

In addition to the DATEX II standard mentioned in the previous chapter, the Mobilithek will continue to support the "container format" introduced with the MDM. This format is based on XML for the transmission of data. It was created specifically for data exchange via the MDM and will continue to be supported by the Mobilithek. The schema of the data format can be found in the container format specification [MCS]. In addition to the actual user data, which is contained in a body element, the data format allows further structural information to be transmitted in a header element, which is used in particular to control the communication process.

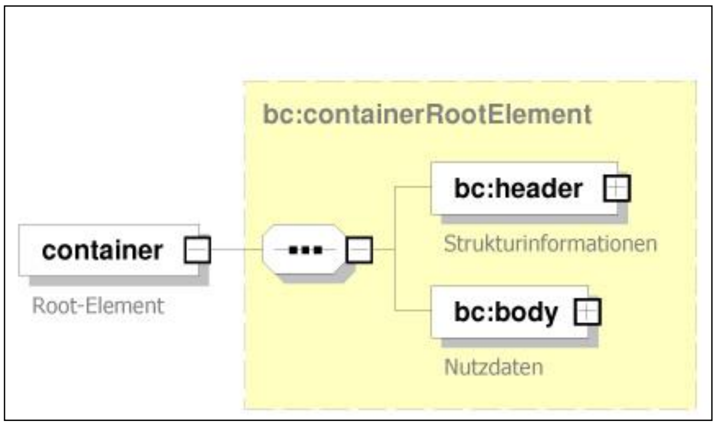


Figure 2: Overview container format

To keep the model flexible, the format and content of the body element are not specified. This means that not only data in XML format can be transported in the container, but also binary data.

3.3 Other data formats

Since Mobilithek passes on the transmitted "payload" unchanged and does not carry out any content checks or evaluations, it is also possible to transmit any data formats via the HTTP/REST interface of Mobilithek in the HTTP body. The payload contained in the HTTP body is passed on unchanged to the data recipient.

4 Interfaces of the broker system of the Mobilithek

The broker system of the Mobilithek assumes the role of the client or the role of the server as an intermediary between the data provider system and the data recipient system, depending on the situation:

The broker system can request data from the data provider as a client or the data provider can send the data to the broker system by itself.

The data recipient can in turn request data from the broker system as a client or the broker system can send the data to the data recipient on its own.

Figure 3 shows the possible paths available for data packet transmission between the data provider and the broker system on the one hand and the broker system and the data recipient on the other.

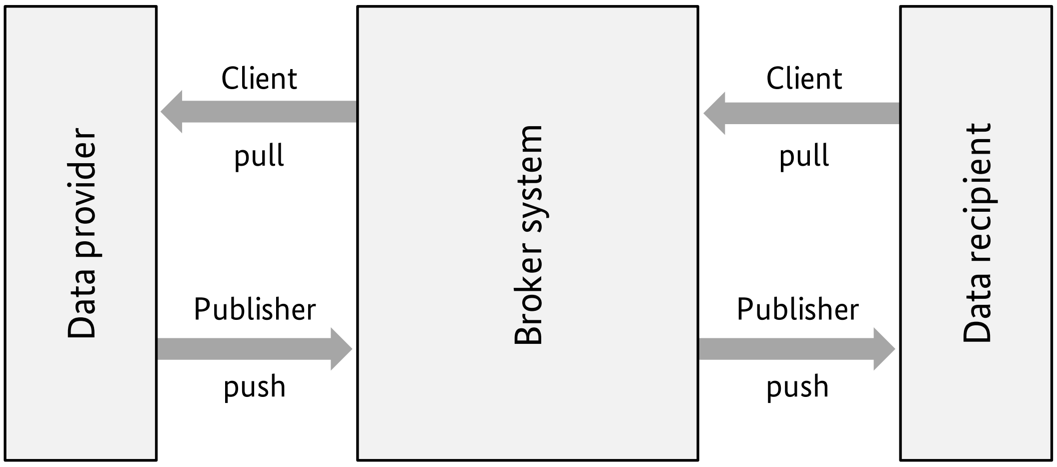


Figure 3: Interfaces between data provider, broker system and data recipient

The data packets received or sent by the broker system must be in DATEX II, container format or "any" payload.

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

Table 6 shows which communication channels are supported. For each data format (DATEX II / Container), communication pattern (Client Pull / Publisher Push) and protocol (HTTPS, SOAP, OCIT-C), if supported, the chapter is noted in which the corresponding communication path is described - differentiated according to data provider and data recipient systems.

In addition, it is indicated in each case whether the data provider or data recipient system acts as a client or as a server vis-à-vis the Mobilithek. Client means here that the system makes requests to the Mobilithek or actively establishes a connection to it.

Server, on the other hand, means that the system is addressed by the Mobilithek and must answer its requests. In this case, network access to the system to be connected must be permitted from the outside (by the Mobilithek).

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

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

In principle, data packages are only available for collection by data recipients during the validity period stored in the publication definition.

As long as no new data package is delivered, the data receiver system receives an error message of the type "No Content", which varies depending on the protocol. More details are described in the protocols.

Important notes on the SOAP protocol endpoints

1. For the SOAP endpoints, the Mobilithek publishes the corresponding WSDLs that specify the endpoints. For reasons of compatibility with the MDM, the Mobilithek also accepts data packets if they do not conform to the payload scheme specified in the WSDL. This means that the Mobilithek does not perform a formal validation against the WSDL. Rather, it limits the formal validation of requests to basic requirements, such as providing a well-formed XML and a valid SOAP request (SOAP Envelope and SOAP Body is included), as well as other minimal requirements to successfully process a request. The specific requirements including the error code are explicitly specified in the respective protocol-specific chapters.
2. As a consequence of point 1, data recipients cannot rely on the delivered data package and thus the SOAP body being compliant with the WSDL.

4.1 Encryption of communication

Data provider systems and data recipient systems can access the services of the platform via the interfaces offered by the Mobilithek. These services for data collection and delivery are offered under defined, standardized URLs [URL] and require certificate-based client authentication via HTTPS [HTTPS]. X.509-compliant certificates are used for client authentication [PKI], these are issued by the operator of the Mobilithek.

In cases where the Mobilithek acts as a client, it uses an X.509 compliant certificate to authenticate itself - if necessary - to the data provider or data recipient system acting as a server, see chapter 10.4

4.2 Compression

Both GZIP-encoded (i.e., compressed) and uncompressed HTTPS requests and responses can be used for data transmission between the Mobilithek and data provider systems.

The data transfer between Mobilithek and data recipient systems is - in deviation to [DATEXIIv2PSM] - always takes place by means of GZIP-encoded HTTPS requests and responses.

This applies to HTTP, SOAP and OCIT-C regardless of the Exchange protocol selected.

4.3 Support of "Delta" data packets

Mobilithek generally supports the processing of Delta data packets.

If the Mobilithek receives a complete data packet, it replaces all data packets currently in the associated data packet buffer. The Mobilithek stores a received "delta" data packet in the sequence of its receipt in the associated data packet buffer.

Data receiver systems can access all existing data packages incl. the last complete delivery in the order of delivery via the data receiver pull interface. This mechanism can also be used by data receivers who have configured PUSH as the delivery mode.

In principle, data users who only access data services via PULL can also benefit from the bandwidth reduction resulting from the use of delta deliveries.

Since the Mobilithek does not evaluate, modify in any respect, or assemble the package content, it is the responsibility of the data provider system to deliver the data packages to the Mobilithek in the order that will allow a data receiver system to assemble them into a complete publication.

The identification whether it is a complete or a delta data package depends on the data format. Delta delivery is currently supported for DATEX II v3.

4.4 Validity period of data packets

A data provider can optionally specify a validity period for a publication via the user interface. If a validity period is specified, it defines the maximum time span between the submission of data packages for the publication in question. If this period is exceeded, the associated packet buffer is marked as "out of date".

This has the following consequences in detail:

- The packet buffer is emptied.
- No further data packets for this data offer are transmitted to data recipients. Ongoing data transmissions are not interrupted.
- If delta support is activated, delta packets are discarded and not forwarded to data recipients.
- From the data recipient's point of view, the situation is as if no data packets are available for this data offer.
- The "out-of-date" status is cancelled when a complete data package is provided again by the data provider.

4.5 Promise of immutability

The Mobilithek is designed to pass on the data delivered by the data provider unchanged to the data receiver(s). The broker system may not change the user data portion, the "DATEX II payload" of the data packets received.

The phrase "promise of immutability" has become established for this principle.

A significant effect of the "immutability promise" is that all namespace declarations that refer to the DATEX II payload must be defined within the <d2LogicalModel> element (DATEX II v2) or at the so-called messageContainer (DATEX II v3) so that these declarations remain part of the payload, e.g. even when delivered via SOAP and subsequently forwarded via HTTP (the SOAP envelope is removed in these cases). If data packages are provided in which namespace declarations are in the SOAP envelope, declarations which are required will be moved to the corresponding XML elements upon delivery of the data package. Further, due to technical reasons a re-sorting of namespace declarations can happen. Those modifications do not change the data package from a semantic or content point of view.

The following is an example for each of DATEX II v2 and DATEX II v3 of an implementation that adheres to the immutability promise.

4.5.1 DATEX II v2

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

4.5.2 DATEX II v3

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

Example for the case, that the namespace declarations have been allocated to the SOAP envelope upon package delivery. Those declarations are now associated with the corresponding XML elements:

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

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

Important notes on DATEX II v3

1. The <codedExchangeProtocol> element (required element), depends on the in/delivery protocol used as well as the type of data packet.
2. When submitting a data package to the Mobilithek, the following value is expected:
    a. "snapshotPush" if it is a complete data packet and the data provider delivers the packet using push.
    b. "deltaPush" if it is a delta data packet and the data provider delivers the packet via push.
    c. "snapshotPull" if the Mobilithek picks up the data packet from the data provider system and it is a complete data packet.
    d. "deltaPull" if the Mobilithek fetches the data packet from the data provider system and it is a delta data packet.
3. Each time the Mobilithek is delivered to the data receiver system, the value of this element is set as follows:
    a. "snapshotPush" if the Mobilithek delivers the data packet to the data receiver using push and the data packet was delivered by the data provider system using either "snapshotPull" or "snapshotPush".
    b. "deltaPush" if the Mobilithek delivers the data packet to the data receiver using push and the data packet was delivered by the data provider system using either "deltaPull" or "deltaPush".
    c. "snapshotPull" if the data receiver sends a pull request to the Mobilithek and the data packet was delivered by the data provider system using either "snapshotPull" or "snapshotPush".
    d. "deltaPull" if the data receiver sends an pull request to the Mobilithek and the data packet was delivered by the data provider system using either "deltaPull" or "deltaPush".
4. SOAP pull requests always deliver only the last complete data package supplied by the data provider. Delta data packets cannot be obtained from the Mobilithek using SOAP-Pull.
5. The value of the <messageGenerationTimestamp> element is not reset when the mobile library is delivered to the data receiver system. This means that the timestamp is an end-to-end value that corresponds to the original timestamp of the delivered data package.

4.6 Use of the interfaces

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

Mode	Description
Client Pull	The communication is initiated by the client (Mobilithek to data provider or data recipient system to Mobilithek) and the data is sent as a response.
Publisher Push Periodic	The communication is initiated by the publisher (data provider system to Mobilithek) at predefined intervals.
Publisher Push on Occurrence	Communication is initiated by the publisher (data provider system to Mobilithek or Mobilithek to data recipient) whenever the data changes.

Table 7: Operating modes of r Mobilithek

For data exchange with the Mobilithek, transport encryption[1] with TLS 1.2 or TLS 1.3 and authentication by means of standard-compliant X.509v3 certificates must be used for all protocols. If the standard protocols provide for basic authentication by means of username and password, these protocol elements are ignored. This applies in particular to the OCIT protocol [OCIT-C]protocol, as will be explained in the following.

The Mobilithek implements an OCIT-C interface based on the OCIT-C standard in version 1.1_R1 of 30.10.2014. The OCIT-C range of functions is only offered by the Mobilithek to a limited extent and under the specification of a specific use of protocol elements. Only DATEX II v2 is used as the data model for OCIT-C as described in this document or in [DATEXIIv2Spec] described. The OCIT-C data models are not supported.

4.6.1 Data provider side

The Mobilithek acts as a subscriber to the data provider (the publisher) and receives the data packets. Depending on the procedure, the broker system can take on the role of a server or client.

[1] The following cipher suites are supported:
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

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

4.6.2 Data recipient side

The Mobilithek acts as a publisher towards the data recipient (the subscriber) and provides the data packets. Depending on the procedure, the broker system can take on the role of a server or client.

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

4.7 Error handling

4.7.1 Client pull from data provider

If the data provider has configured Pull as the communication path and the data provider system responds to a Pull Request from the Mobilithek with an error, the Pull Request will not be repeated immediately. The Mobilithek will execute the next pull request according to the time period specified by the data provider.

4.7.2 Publisher Push

If the data recipient has configured push as the communication path and the configured data recipient system cannot be reached, the Mobilithek will attempt to reach the data recipient system at increasingly longer intervals. Only when the data receiver system is technically reachable will a new data delivery be attempted. The interval between the intervals grows exponentially. The technical implementation is protocol-specific and is explained in the corresponding chapters.

If the data receiver system responds to a transmission attempt with an HTTP response status of 500 or 4xx, the transmission attempt is repeated immediately. If this transmission attempt also fails, a new data transmission is only initiated by the Mobilithek after receipt of the next complete data delivery by the data provider.

4.8 Use of the "If-Modified-Since" header field in the HTTPS protocol

The broker system supports the header field "If-Modified-Since" in connection with the field "Last Modified" (cf. [HTTP/1.1]). This avoids the repeated sending or collection of messages that have already been delivered. According to the HTTP standard, the resolution of the time stamp is on the second level. The Mobilithek rounds up the timestamp, which is set when the data packet is received, to the next full second.

If the HTTP request does not contain the header field "If-Modified-Since", the last data packet delivered will be delivered by the Mobilithek.

In connection with publications for which delta support is activated, data recipients can also benefit from the bandwidth reduction made possible by the use of delta data packets via the PULL interfaces by using this header. For this purpose, the first PULL request is started with a timestamp far in the past. This request delivers the oldest data packet from the data buffer, which by definition is a complete data packet (see chapter 4.3). In the following PULL requests, the timestamp from the header element "Last-Modified" is used.

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 is requested with a request containing the following header line:

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

4.8.1 Data provider

The broker system always sends the requests with an "If-Modified-Since" header field if the data provider system had set the "Last-Modified" header field in its response.

The data provider system should always set this header field to enable the Mobilithek to apply this feature!

4.8.2 Data recipient

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

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

4.8.3 Unchanged data

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

5 Data format-independent interfaces

5.1 Deleting the publication content

Via this interface, a data provider system can request the Mobilithek to delete all contents of a publication. The publication itself remains intact.

Request to the Mobilithek

The data provider system sends an HTTPS DELETE request to the mobile library. The associated packet buffer is identified with the publication ID passed in the request.

The URL of the broker system is structured as follows:

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

Response to the data provider

The Mobilithek broker system generates an HTTPS response after receiving the request. If the request was processed successfully, the Mobilithek broker system responds with a response code 200 and an empty response body.

In the event of an error, the response body contains an error text.

Status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the Table 18 described meanings apply:

Description	
Request	Request DELETE /mobilithek/api/v1.0/publication/2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] ▪ 400: Incorrect request, e.g. specification of a non-numeric publication ID. ▪ 403: The data provider system is not authorised to delete the content of the specified publication. ▪ 404: Publication was not found or no publication ID was specified. ▪ 503: Service Unavailable (e.g. during maintenance)

Table 8: Request/response between Mobilithek/data provider system when deleting the publication content

6 DATEX II v2

6.1 SOAP interface

6.1.1 Data provider side

6.1.1.1 Client Pull SOAP

In the Client Pull SOAP exchange procedure, the Mobilithek broker system requests the data provider system to deliver its data to the Mobilithek.

Providing a web service

The data provider system must offer a web service with the method getDatex2Data, which is defined based on the DATEX II Pull WSDL [DATEXIIv2Pull] is defined. As input nothing is expected, as output the broker system of the mobile library gets back the requested data in DATEX II format: in the body element an object of the type d2LogicalModel is expected. According to the note in chapter 4 the Mobilithek broker system accepts all data packets, provided they are provided in the form of a valid XML. The broker system accepts the content of the SOAP body as a data package.

Via the administration component of the Mobilithek, the data provider must store the URL of his service endpoint in the publication configuration.

Calling a web service

The broker system of the Mobilithek provides a web service client defined based on the DATEX II Pull WSDL [DATEXIIv2Pull] for calling web services. This web service must return data according to a DATEX II v2 schema [DATEXIIv2Schema]. It is expected that a suitable profile will be used from the overall schema.

The broker system identifies the data provider systems that have subscribed to a pull procedure and the associated service endpoints in the metadata directory and calls them cyclically according to the configured publication frequency. The data received after the call are temporarily stored in corresponding packet buffers for delivery to potential data recipients. Any previous data package that may still exist is replaced in the process.

6.1.1.2 Publisher Push SOAP

With the Publisher Push exchange procedure, the data provider system must deliver the data to the Mobilithek on its own. A corresponding SOAP interface must be used. Whether the data is generated because of an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant for the functioning of the Mobilithek broker system. The exchange mechanism is identical in both cases.

Providing a web service

The broker system of the Mobilithek offers a web service with the method putDatex2Data, which is defined based on the specification DATEX II Push WSDL [DATEXIIv2Push] specification. The data to be transferred is expected as input, and the data provider system receives confirmation data in DATEX II format as output. An object of the type d2LogicalModel is expected in the body element.

The output consists of an acknowledgement of receipt (Acknowledge message, see below).

In the URL of the service end point at the broker system, the ID of the publication in which the data packets are to be placed must be entered.

The URL is structured as follows:

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

If a numeric publication ID has been specified but does not exist or is not active, the Mobilithek responds with an HTTP response code 200 and a SOAP response with the denyReason "wrongCatalogue" and the response code "requestDenied":

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

If the DATEX II element in the <exchange> element" contains a <keepAlive> element with the value true, no payload is expected. The mobile library responds with an acknowledge message.

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

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

Calling the web service

The data provider system must provide a web service client defined based on the DATEX II Snapshot Push WSDL [DATEXIIv2Push] to invoke the web service. The web service must deliver the data to the publication-specific service endpoint of the broker system of the mobile library. The URL of the service endpoint to be used is displayed in the publication configuration of the Mobilithek administration component when the publication is created. The Mobilithek broker system accepts this data and stores it in a packet buffer. Any previous data package that may still exist is replaced in the process.

Error codes

HTTP Response Code	SOAP Response	Description
200	denyReason: requestDenied response: unknownReason	Invalid XML or invalid SOAP request; error when unpacking the request.
		the publication is not a DATEX2 V2 publication or the protocol specified by the data provider is not PUSH SOAP.
	denyReason: requestDenied response: wrongCatalogue	There is no active publication with the specified publication ID
400	empty body	The specified publication ID is not numeric.
403	empty body	The user assigned to the machine account is not authorised to provide data for the specified publication.
404	empty body	No publication ID specified

Table 9: Error codes for DATEX II V2 - Publisher Push SOAP

6.1.2 Data recipient side

6.1.2.1 Client Pull SOAP

In the client pull SOAP exchange method, the data receiver system must request the mobile library to send data to the data receiver system.

Providing a web service

The broker system of the Mobilithek provides a web service with the method getDatex2Data, which is defined based on the specification [DATEXIIv2Pull] is defined. The subscription ID is expected as input in the URL, and the data recipient receives the requested data in DATEX II format as output. An object of the type d2LogicalModel is expected in the body element. Based on the transmitted subscription ID, the Mobilithek can determine the associated package buffer and the data package.

Example:

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

Calling the web service

The data receiving system must provide a web service client to [DATEXIIv2Pull] specification to call the web service. The corresponding subscription ID must be included in the URL as an input parameter.

The SOAP endpoint of the broker system is:

mobilithek

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

Error codes

HTTP Response Code	SOAP Response	Description
200	faultcode: Server faultstring: Datex-II ClientPull - no data	The associated packet buffer does not contain a data packet
400	empty body	The specified subscription ID is not numeric
		Required HTTP headers are missing, e.g. "Accept-encoding" header is not specified.
		HTTPS Request Body is empty
404	empty body	No subscription ID specified
406	empty body	In the accept-encoding header, gzip is missing as an accepted encoding.
500	faultcode: Server faultstring: Contract can not be found, is not active or not available for provided orgId	The user assigned to the machine account is not authorised to obtain data under this subscription or the specified subscription does not exist.
	faultcode:Server faultString: Offer validation not passed reason: Access protocol, data model don't match	The publication associated with the subscription is not a DATEX2 V2 publication.

Table 10: Error codes for data receiver DATEX II V2 Pull SOAP

6.1.2.2 Publisher Push SOAP

In the Publisher Push exchange procedure, the broker system of the Mobilithek delivers the data to the data recipient systems on its own. A corresponding SOAP interface is used for this. Whether the data is generated based on an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant, the mechanism for delivery to the data recipient is identical.

Providing a web service

The data receiver system must offer a web service with the method putDatex2Data, which is defined based on the specification [DATEXIIv2Push]. The requested data is expected as input, and the mobile library receives confirmation data in DATEX II format as output. An object of the type d2LogicalModel is expected in the body element.

Calling the web service

The broker system provides a web service client based on [DATEXIIv2Push] for calling up the data recipient web services. Via the administration component of the Mobilithek, the data recipient must store his service endpoint in the subscription configuration.

The broker system identifies these data recipient systems and starts a corresponding web service call.

If the transmission of the data could be completed successfully, the broker system expects a - corresponding acknowledgement message from the data receiver system. The following example shows the content of such a so-called acknowledgement response, which is contained in the body element when transmitting with the SOAP protocol.

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

If the data receiving system does not acknowledge the transmission with an acknowledge response, the transmission is repeated in accordance with chapter 4.7.2 the transmission is repeated.

If the data receiving system is technically not accessible, the mobile library sends "keep-alive" messages to the data receiving system until it responds with an acknowledgement response. Only then is the transmission of data continued. The example below shows a "keep-alive" message:

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

6.2 HTTPS interface

6.2.1 Data provider side

6.2.1.1 Client Pull HTTPS

In the client pull exchange procedure, the broker system of the Mobilithek cyclically requests the data provider system to deliver its data to the Mobilithek. The time interval used must be configured when configuring the data offering in the metadata directory. For this exchange, the following points apply from the Simple HTTP Server Profile of the [DATEXIIv2PSM], Chapter 4, points C.1-C.12.

It should be noted that the other 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 procedure, which is mandatory for the Mobilithek. C. 18-C. 27 are not applicable, as the options only relate to the optional provision of DATEX II data in file form, which is not used for the Mobilithek. Return values are defined differently from C.15. The rule for building URLs to the payload (C.16) is not applied. See also Appendix B - DATEX II HTTP Protocol Support.

Request to the data provider

The Mobilithek broker system sends an HTTPS GET request to the data provider system from which the data is to be fetched. The Mobilithek is able to identify data provider systems that have subscribed to a pull procedure and send requests to them at defined intervals.

The data provider must store the publication-specific server URL in the publication configuration via the administration component of the Mobilithek. The URL must be stored in full by the data provider. The Mobilithek does not add parameters to it, such as the publication ID.

Please also note the instructions in chapter 4.8 “Use of the "If-Modified-Since" header field”.

Response to the Mobilithek

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

The broker system of the Mobilithek accepts this data and stores it in a packet buffer. Any previous data package that may still be present is replaced in the process.

6.2.1.2 Publisher Push HTTPS

The data provider system must send a data packet for a publication to the Mobilithek broker system.

Request to the Mobilithek broker system

The data provider system must send an HTTPS POST request with a message in the HTTP request body to the Mobilithek broker system.

The publication ID must be specified as a path element in the URL. The user data is passed in the HTTP request body

The Content-Type header to be sent depends on the syntax to which the corresponding data offer is set.

The URL of the broker system is structured as follows:

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

It should be noted that the HTTPS interface does not support "keep-alive" messages.

Response to the data provider

The data provider system receives an HTTPS response to the request. The message body is empty, the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings in Table 25 apply.

Description	
Request	Request POST /mobilithek/api/v1.0/publication/datexv2/<publicationID> HTTP/1.1 Host: mobilithek.info Content-Type: text/xml oder application/xml or application/json Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] The following status codes have a special meaning: ▪ 400: The specified publicationId is not numeric or the request is not structured correctly. ▪ 404: Publication parameter could not be assigned, the publication is no longer valid or no value was specified after the slash in the URL path ▪ 403: The user is not authorized to submit data via this endpoint for the specified publication, or the publication has not been configured to be delivered over HTTPS.

Table 11: Request/Response between Provider System/Mobilithek during Publisher Push DatexIIv2 HTTPS

6.2.2 Data recipient side

6.2.2.1 Client Pull HTTPS

In the client pull exchange procedure, the data receiver system must request the broker system of the mobile library to transmit the data.

Request to the Mobilithek

The data receiver system must send an HTTPS GET request to the URL of the mobile library. Based on the subscription ID, the associated packet buffer as well as the data packet is determined.

The subscription ID must be passed in the path of the URL and additionally as a parameter. The URL of the broker system is therefore structured as follows:

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

Please also note the instructions in chapter 4.8 “Use of the "If-Modified-Since" header field”.

Response to the data recipient

The broker system of the Mobilithek generates an HTTPS response after receiving the request. For this purpose, the corresponding packet buffer and the appropriate data packet are determined based on the subscription ID. The content of the data packet is transmitted to the data recipient in the body of the response. According to the DATEX II Client Pull HTTP Profile [DATEXIIv2PSM] Chapter 4, the response has the content type "text/xml; charset=utf-8" and is - in deviation to [DATEXIIv2PSM] - is always sent in GZIP compressed form.

Status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings Table 12 described meanings apply.

Description	
Request	Request GET /mobilithek/api/v1.0/subscription/2000000/clientPullService?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <d2LogicalModel > ... </d2LogicalModel >
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 204: No data packet in the packet buffer for subscription. ▪ 304: No data packet in the packet buffer that is younger than the timestamp in the "if-modified-since" header. ▪ 400: No subscription parameter specified in the request or missing "Accept-Encoding" header or subscription parameter is not numeric. ▪ 403: The user is not authorised to retrieve this subscription via this endpoint or it is not a DATEX2 V2 publication associated with this subscription. ▪ 404: Subscription is no longer valid or does not exist. ▪ 406: gzip not specified in the "Accept-Encoding" request header.

Table 12: Request/Response between Mobilithek/Data Recipient System with Client Pull HTTPS

6.3 OCIT-C interface

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

6.3.1 Scope of functions

The Mobilithek implements the subset of protocol functions from the functional scope of the OCIT-C standard that are required for the transmission of a current data package with all the information of a publication. The exchange of subsets of data (delta deliveries) is not supported. Historical data can also not be queried.

The Mobilithek implements a web service with the full WSDL OCIT_Cif.wsdl, which is accessible under specific OCIT endpoints. However, calling an unsupported operation is answered with an unmodelled SOAP fault (HTTP response code 500) with the value "Method not found".

The data schema is defined by the OCIT-C schema protocol.xsd. The OCIT messages use a data list to transport the data, which can contain several data objects. In communication with the Mobilithek, the data list may only ever contain exactly one data object. The DATEX II package must be transparently embedded in the <data> element of the message. Data deliveries with multiple packages are acknowledged with an error.

The <data> element of the OCIT-C message is specified in the protocol.xsd as an element of the type *anyType*. For SOAP-compliant transmission, the <data> element must be typed. For this purpose, a new data type anyD2LogicalModel is introduced using the OcitCDatex2.xsd below.

```xml
<?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 scheme can be referenced via the following URL:
https://bast.s3.amazonaws.com/schema/1446644360562/OcitCDatex2.xsd

6.3.2 Data provider side - Publisher Push OCIT-C

The Publisher Push functionality is mapped to the OCIT-C method put. A put call must always be uniquely assigned to a publication by referencing a publication ID. This publication ID, which is automatically assigned by the MDV of the mobile library, must be transferred by the data provider system in the OCIT-C element <objectType>.

A put message must contain exactly one element of the DATEX II type D2LogicalModel. For this purpose, the request must contain a data list with exactly one data object. A request with several data objects is rejected by the Mobilithek with an error. The delivery of a DATEX II element must always be complete, i.e. contain all data points or objects of the publication. However, Mobilithek will not check this. It is the responsibility of the data provider system to ensure completeness.

In the metadata management of the Mobilithek, the DATEX II element can be manually validated against the publication schema stored in the Mobilithek. This schema may only describe the DATEX II payload without the OCIT-C container. Validation of the entire OCIT message does not take place.

The following paragraph shows an example of a possible delivery in OCIT-C format for a publication with the fictitious ID=2600103 of a fictitious organisation "TEST". The DATEX II payload is shown in abbreviated form.

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

During data delivery, the Mobilithek ignores the following elements from the OCIT-C protocol:

- username
- password
- identifier within the putds attribute

The Mobilithek acknowledges the delivery with an OCIT message of the type *putResponse*. The elements are set as follows:

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

The following paragraph shows an example 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>
```

Error codes

With the exception of the situation where an invalid SOAP request is made, the Mobilithek responds to errors with an HTTP Response 200 and a corresponding error code and text. The following table describes the error codes and situations.

HTTP Response Code	SOAP Response	Description
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalid XML or invalid SOAP request
	SOAP Fault	Multiple SOAP methods identified in the request, or

HTTP Response Code	SOAP Response	Description
	faultcode: Client faultstring: SOAP action cannot be determined	the first element in the SOAP body is not one of the supported SOAP methods: <put>, <inquireAll>, <get> or <waitForGet>.
200	errorCode: 1 errorText: access error - erroneous object type	The specified publication ID is not numeric
	errorCode: 1 errorText: access error - exactly one putds must be present	The request contains more than one <putds> element
	errorCode: 14 errorText: found empty object type	No publication ID found in <objectType> element
	errorCode: 15 errorText: object type not found - is missing	<objectType> Element not found in the request.
	errorCode: 1 errorText: access error	The user assigned to the machine account is not authorised to provide data for the specified publication, or the specified publication is not a DATEX2 V2 publication, or OCIT-C has not been defined as an access protocol.
	errorCode: 1 errorText: access error - no valid certificate-publication match	The specified publication ID does not exist.
	errorCode: 1 errorText: access error - exactly one putds must be present	The SOAP request supports exactly one <putds> element

Table 13: Error codes for OCIT-C Publisher Push

6.3.3 Data receiver side - Client Pull OCIT-C

The Client Pull functionality is mapped to the following three OCIT-C methods:

- inquireAll
- get
- wait4Get

An OCIT-C client can synchronise itself to the current data status after it has been started using the inquireAll method. The Mobilithek supports the inquireAll method for this purpose. In the inquireAllResponse, the Mobilithek passes the last valid packet and an internal ID to the client. The client can then use the get or wait4Get methods to continuously fetch current packets. In doing so, the client must always refer to its last package ID. If there is no new package in the mobile library, the get method returns immediately with an empty response. The wait4Get method waits until a current data package is available or a maximum timeout specified by the client or defined by the server has been reached. By using the wait4Get method, a push characteristic can be implemented in the direction of the data recipient. In contrast to the actual OCIT-C behaviour, the Mobilithek always returns a complete data packet with a get or wait4Get response and not only delta data related to the last position. The Mobilithek does not support delta packets for DATEX II v2.

As an alternative to an inquireAll call, a client can also call the get method with the element value position=0 in order to initialise itself or to fetch the last available packet in this way at any time.

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

- username
- password
- watchdog

The attribute filterList in the call is also not supported in all three methods and must always be requested empty by the data receiver system.

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

The following paragraph shows an example of a request for delivery in OCIT-C format for a fictitious subscription with ID=2871015 of a fictitious organisation "TEST".

```xml
<?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. The Mobilithek sets the following OCIT-C elements as follows:

- lastStart = an undefined constant time that the client should ignore.
- errorCode = 0
- errorText = without content
- storetime/tstore = time of delivery of the publication in the Mobilithek
- position = packet ID, ID of the current data packet, only relevant for the OCIT-C methods specified here.
- objectState = modified
- identical = None
- data = DATEX II Payload

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

With the help of the <position> element from the inquireAllResponse, the data receiver system can parameterise the get or wait4Get method in the following to read subsequent packets.

A get call must always be uniquely assigned to a subscription by referencing a subscription ID and to a data package by referencing the package ID. The data receiver system must pass this subscription ID in the OCIT-C attribute <objectType>, the package ID in the attribute <position>. A get call using start and end time is not supported by the Mobilithek.

The following example shows a get request for delivery in OCIT-C format for a fictitious subscription with ID=2871015 and fictitious predecessor package ID=3876098:

```xml
<?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 Mobilithek then forms the getResponse and analogously a wait4GetResponse using the same attributes as in the inquireAllResponse.

The same requirements apply to the wait4Get call as to the regular get call. In addition, the data receiver system should transmit the timeout value of the client in the <maxWaitTime> element. If this element is not transmitted, the wait4Get call behaves like a regular get call and returns immediately with an empty response if no new data packet has been delivered in the meantime. If this value is above the maximum value of 120 seconds configured in the Mobilithek, the Mobilithek default timeout is applied and the caller receives a response after 120 seconds at the latest. This is empty if no new data packet has been delivered within the waiting time.

The possibility of reading different objects with a single wait4Get call is not supported by the Mobilithek. Therefore, only one subscription can be queried at a time with a wait4Get call. List queries are rejected with an error.

The following paragraph shows an example of a wait4Get request for delivery in OCIT-C format for a fictitious subscription with the ID=2871015, the fictitious content ID=3876098 and the value 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>
```

Below is an example wait4Get response from Mobilithek:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
<soapenv:Header/>
  <soapenv:Body>
    <ocit:wait4GetResponse xmlns:ocit="http://odg_und_partner/OCIT_C">
    <ocit:lastStart>2021-07-22T15:37:38.000+02:00</ocit:lastStart>
    <ocit:errorCode>0</ocit:errorCode>
    <ocit:errorText></ocit:errorText>
    <ocit:waitResponseList>
      <ocit:storetime>2021-08-02T11:53:51.480+02:00</ocit:storetime>
      <ocit:objectType>0</ocit:objectType>
      <ocit:position>0</ocit:position>
      <ocit:dataList>
        <ocit:ds>
          <ocit:tstore>2021-08-02T11:53:51.480+02:00</ocit:tstore>
          <ocit:objectState>modified</ocit:objectState>
          <ocit:identifier>
            <ocit:ident>None</ocit:ident>
          </ocit:identifier>
          <ocit:data xsi:type="ns1:anyD2LogicalModel"
                     xmlns:ns1="http://odg_und_partner/OCIT_C/Datex"
                     xsi:schemaLocation="http://odg_und_partner/OCIT_C/Datex
                                         http://bast.s3.amazonaws.com/schema/
                                         1446644360562/OcitCDatex2.xsd">
          </ocit:data>
        </ocit:ds>
      </ocit:dataList>
    </ocit:waitResponseList>
    </ocit:wait4GetResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Data packets are always delivered to the Mobilithek in compressed form. GZIP compression is used for this. This also applies to delivery with the OCIT-C protocol. Data recipient systems must therefore decompress the packages in the web server before they can be processed further with the help of the OCIT-C protocol.

Error codes

With the exception of the situation where an invalid SOAP request is made, the Mobilithek responds to errors with an HTTP Response 200 and a corresponding error code and text. The following table describes the error codes and situations.

HTTP Response Code	SOAP Response	Description
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalid XML or invalid SOAP request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Multiple SOAP methods identified in the request, or the first element in the SOAP body is not one of the supported SOAP methods: <put>, <inquireAll>, <get> or <waitForGet>.
	errorCode: 0 No data	No data packets available, or no data packets available according to the <position> element specified in the request
400	empty body	missing "Accept-Encoding" header or subscription parameter is not numeric
406	empty body	In the accept-encoding header, gzip is missing as an accepted encoding.
200	errorCode: 1 errorText: access error - erroneous object type	The specified subscription ID is not numeric
	errorCode: 14 errorText: found empty object type	No subscription ID found in <objectType> element
	errorCode: 15 errorText: object type not found - is missing	<objectType> Element not found in the request.
	errorCode: 1 errorText: access error - no valid certificate-subscription match	The user assigned to the machine account is not authorised to obtain data for the specified subscription, or the specified subscription is not a DATEX2 V2 publication
		The specified subscription ID does not exist.

HTTP Response Code	SOAP Response	Description
	errorCode: 1 errorText: access error - exactly one position must be present	The call of the <get> or <wait4Get> request requires the specification of the <position> element with a numerical value. This has not been specified according to WSDL.

Table 14: Error codes for OCIT-C data receiver pull

7 DATEX II v3

In contrast to DATEX II v2 Exchange, data transport in DATEX II v3 Exchange 2020 takes place via a MessageContainer structure. Since not all elements of this structure that are possible according to the DATEX II v3 specification are used in the Mobilithek, we speak here of a minimum MessageContainer. In addition to a payload element, this must also contain an <exchangeInformation> element. The minimal MessageContainer as well as the <exchangeInformation> element are available under [DATEXIIv3Exc] available.

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

- <exchangeContext>:
  - <codedExchangeprotocol>: An attribute of an enumeration type. The value differs depending on the protocol used (see also chapter 4.5.2):
    - For SOAP interfaces, either "snapshotPull", "snapshotPush", "deltaPull" or "deltaPush" is used here.
    - For HTTP pull, "snapshotPull" or "deltaPull" is used here.
  - <exchangeSpecificationVersion>: The Mobilithek expects the value "3.0" here.
  - <supplierOrCisRequester>: To be compliant with the standard, an empty XML element must be included here.
- <dynamicInformation>:
  - <exchangeStatus>: The value "online" is expected here.
  - <messageGenerationTimestamp>: current time of message generation.

The <codedExchangeprotocol> element thus indicates whether a DATEX II v3 data packet is a full or a delta packet. The handling of delta packets is described in chapter 4.3 is described.

In the administration component of the Mobilithek, the data provider can select whether delivery of delta packages is planned for this publication.

Important note

If delta packages are delivered for a DATEX II v3 publication, data recipients cannot fetch them via SOAP using a pull request. SOAP pull requests will always deliver the last complete data package delivered.

7.1 Notes on handling schemas with Exchange 2020

Exchange 2020 supports the delivery of publications in XML or JSON format. If one wants to set up a publication for DATEX II v3 content, the data provider must provide several schemas (XML or JSON schema) on two levels:

1. **Content data:** DATEX II v3 has introduced the concept of namespaces in DATEX II. When creating the data profile for a publication, a separate schema is created for each namespace. Each instance of the publication must reference the entry schema, which is called either DATEXII_3_D2Payload.[xsd|json] (Level A or B) or LevelC_3_D2Payload.[xsd|json] (Level C), depending on the compatibility level. This schema imports all other schemas of the respective data profile.
2. **Protocol data:** To transport DATEX II v3 content with the corresponding DATEX II Exchange 2020 specification, the content data schema is embedded in three additional schemas for the options implemented on the Mobilithek: MessageContainer.[xsd|json], InformationManagement.[xsd | json] and ExchangeInformation.[xsd | json].

The protocol data schemas have been profiled for use in the context of the Mobilithek. It is important to note that the schema of the DATEX II MessageContainer object is where the protocol data is linked to the content data. This schema must therefore be adapted depending on whether Level A or B content is to be transported, or Level C content. The procedure is described below for both cases:

7.1.1 DATEX II v3 Level A or B

When generating the Schemas of the publication's data profile, several schemas are generated. The schema to be embedded in each instance of the publication is called DATEXII_3_D2Payload.[xsd|json]. In XML it defines the namespace http://datex2.eu/schema/3/d2Payload. The data provider must upload these content data schemas together with the variant of ExchangeInformation.[xsd|json] as well as MessageContainer.[xsd|json] for Level A and B in the user interface to his publication [DATEXIIv3Exc].

7.1.2 DATEX II v3 Level C

When generating the Schemas of the publication's data profile, several schemas are generated. The schema to be embedded in the instances of the publication is called LevelC_3_D2Payload.[xsd | json] and defines in XML the namespace http://levelC/schema/3/d2Payload. The data provider must upload these content data schemas together with the variant of ExchangeInformation.[xsd | json] as well as MessageContainer.[xsd | json] for Level C in the user interface to his publication [DATEXIIv3Exc].

Important note

DATEX II v3 content on the Mobilithek must always be based on an entry element derived from the abstract class PayloadPublication in the DATEX II v3 package Common. This is independent of which compatibility level is used, as the Exchange 2020 MessageContainer expects such an object.

The schema generated from the associated DATEX II Package Common must always be part of the content data schemas, i.e. either DATEXII_3_Common.[xsd | json] (Level A or B) or LevelC_3_Common.[xsd | json] (Level C).

Users aiming for a Level C publication with a different structure will need to make appropriate manual adjustments at schema level. If necessary, they should contact Mobilithek Support for assistance.

For technical reasons, referenced subschemas in JSON must start with the file: prefix. For example:

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

7.2 SOAP interface

7.2.1 Data provider side

7.2.1.1 Client Pull SOAP

In the Client Pull SOAP exchange procedure, the Mobilithek broker system requests the data provider system to deliver its data to the Mobilithek.

Providing a web service

The data provider system must offer a web service that is defined based on the DATEX II Snapshot Pull WSDL [DATEXIIv3Pull]. As input nothing is expected, as output the broker system of the mobile library expects the requested data in a MessageContainer in DATEX II format according to the minimum MessageContainer profile in the schema MessageContainer.xsd [DATEXIIv3Exc].

It is the responsibility of the data provider to define the mandatory <exchangeInformation> element with its <exchangeContext> and <dynamicInformation> elements. In order to provide the data in a standard-compliant manner, it should be noted that the <codedExchangeProtocol> element is set to the value

- "snapshotPull" for a complete data package
- "deltaPull" for a delta data package

is to be set.

Irrespective of this, the Mobilithek replaces the <codedExchangeProtocol> element with the respective value corresponding to the delivery protocol (see chapter 4.5.2), in order to enable standard-compliant data receiver systems to process the data correctly. Following the hint in chapter 4 the broker system only carries out necessary validations of the response. In order for the data packet to be accepted by the mobile library, it must have a valid XML format and the <codedExchangeProtocol> element specified above must be assigned a valid value.

Via the administration component of the Mobilithek, the data provider must store the URL of his service end point in the publication configuration at which the Mobilithek is to retrieve the data package.

Calling a web service

The broker system of the Mobilithek provides a web service client defined based on the DATEX II Snapshot Pull WSDL [DATEXIIv3Pull] for calling web services. This web service must return data according to the schema MessageContainer.xsd [DATEXIIv3Exc] schema.

If there is no data packet for a delivery in the data provider system, the broker system of the Mobilithek expects the following response:

- HTTP Response Code 200 Ok

- a minimal MessageContainer element without payload

Example response:

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

The broker system identifies the data provider systems that have subscribed to a pull procedure and the associated service endpoints in the metadata directory and calls them cyclically according to the configured publication frequency. The data received after the call are temporarily stored in corresponding packet buffers for delivery to potential data receivers. If the delivered data package is a complete package, all data packages in the package buffer are replaced. A delta packet is appended to the list of data packets in the packet buffer.

7.2.1.2 Publisher Push SOAP

With the Publisher Push exchange procedure, the data provider system must deliver the data to the Mobilithek on its own. A corresponding SOAP interface must be used. Whether the data is generated based on an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant for the functioning of the Mobilithek broker system. The exchange mechanism is identical in both cases.

Providing a web service

The broker system of the Mobilithek offers a web service that is defined based on the specification DATEX II Snapshot Push WSDL [DATEXIIv3Push] specification. The data to be transferred is expected as input in a MessageContainer instance in the body element of the SOAP envelope.

It is the responsibility of the data provider to define the mandatory <exchangeInformation> element with its <exchangeContext> and <dynamicInformation> elements. In order to provide the data in a standard-compliant manner, it should be noted that the <codedExchangeProtocol> element is set to the value

- "snapshotPull" for a complete data package
- "deltaPull" for a delta data package

is to be set.

Irrespective of this, the Mobilithek replaces the <codedExchangeProtocol> element with the value corresponding to the delivery protocol (see chapter 4.5.2), in order to enable standard-compliant data receiver systems to process the data correctly.

In the URL of the service end point at the broker system, the ID of the publication is entered in which the data packets are to be placed.

The URL is structured as follows:

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

Example:

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

Calling the web service

The data provider system must provide a web service client defined based on the DATEX II Snapshot Push WSDL [DATEXIIv3Push] to call the web service. The web service must deliver the data to the publication-specific service endpoint of the broker system of the mobile library. The broker system accepts this data and stores it in a packet buffer. If the delivered data package is a complete package, all data packages in the package buffer are replaced. A delta packet is appended to the list of data packets in the packet buffer.

A successful delivery is answered by the Mobilithek with a response in the form of a 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 <URL of schema
file>"
        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>
```

However, the Mobilithek responds to a faulty delivery with a response in the form of a DATEX II ExchangeInformation with the negative return status "fail" according to the schema definition in ExchangeInformation.xsd [DATEXIIv3Exc]e.g. if the publication is not configured for the SOAP push procedure. In the response, the following values are used for the <codedInvalidityReason> element:

- invalidPayload if no publication was found for the specified ID.
- invalidMessage if the SOAP request could not be unpacked correctly.

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 <URL of schema
file>"
        modelBaseVersion="3">
  <ex:exchangeContext>
    <ex:codedExchangeProtocol>snapshotPush</ex:codedExchangeProtocol>
    <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
    <ex:supplierOrCisRequester></ex:supplierOrCisRequester>
  </ex:exchangeContext>
  <ex:dynamicInformation>
    <ex:exchangeStatus>online</ex:exchangeStatus>
    <ex:messageGenerationTimestamp>2021-08-06T15:49:33.600+02:00
    </ex:messageGenerationTimestamp>
    <ex:returnInformation>
      <ex:returnStatus>fail</ex:returnStatus>
      <ex:codedInvalidityReason>invalidPayload</ex:codedInvalidityReason>
    </ex:returnInformation>
  </ex:dynamicInformation>
</ex:putSnapshotDataOutput>
```

Error codes

HTTP Response Code	SOAP Response	Description
200	returnStatus: fail codedInvalidityReason:invalidMessage	Invalid XML or invalid SOAP request; error when unpacking the request, or The publication is not a DATEX II V3 publication or the protocol specified by the data provider is not PUSH SOAP.
	returnStatus: fail codedInvalidityReason: invalidPayload	The <codedExchangeProtocol> element was not specified according to WSDL or does not contain the value "snapshotPush" or "deltaPush", or the publication ID does not exist
400	empty body	The specified publication ID is not numeric.
403	empty body	The user assigned to the machine account is not authorised to provide data for the specified publication.
404	empty body	No publication ID specified

Table 15: Error Codes for DATEX2 V3 Publisher Push SOAP

7.2.2 Data recipient side

7.2.2.1 Client Pull SOAP

In the client pull SOAP exchange method, the data receiver system must request the mobile library to send data to the data receiver system.

Providing a web service

The broker system of the Mobilithek offers a web service that is defined based on the specification [DATEXIIv3Pull]. The subscription ID is expected as input in the URL, and the data recipient receives the requested data as output in the payload element of a message container in DATEX II Exchange 2020 format. Based on the transmitted subscription ID, the mobile library can determine the associated packet buffer and the data packet.

Important note

- If the packet buffer does not contain a data packet at the time of the request, the Mobilithek responds to the request with a MessageContainer without a payload element. (cf. Calling a web service)
- Even if delta packets are submitted for publication by the data provider, the SOAP request always returns only the last complete data packet submitted.
- If the data provider has submitted the data to the Mobilithek via the HTTPS interface, it is possible that the data does not have a MessageContainer. This data cannot be retrieved via the SOAP interface as a data recipient and is answered by the Mobilithek with a 500 response code.

Calling the web service

The data receiving system must provide a web service client to [DATEXIIv3Pull] specification to call the web service. The corresponding subscription ID must be included in the URL as an input parameter.

The SOAP endpoint of the broker system is:

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

Example:

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

Error codes

HTTP Response Code	SOAP Response	Description
200	DATEX2 V3 Response without <payload> element	The associated packet buffer does not contain a data packet
405	empty body	The parameter ? subscriptionId= was not specified or no value was specified for the parameter.
406	empty body	In the accept-encoding header, gzip is missing as an accepted encoding.
500	faultcode: Client faultstring: Contract can not be found, is not active or not available for provided orgId	The user assigned to the machine account is not authorised to obtain data under this subscription or the specified subscription does not exist.
	faultcode:Client faultstring: invalid - XML	Invalid XML or invalid SOAP request
	faultcode:Client faultString: Offer validation not passed reason: Access protocol, data model don't match	The publication associated with the subscription is not a DATEX2 V3 publication.

Table 16: Error codes for data receiver DATEX II V3 Pull SOAP

7.2.2.2 Publisher Push SOAP

In the Publisher Push exchange procedure, the broker system of the Mobilithek delivers the data to the data recipient systems on its own. A corresponding SOAP interface is used for this. Whether the data is generated because of an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant, the mechanism for delivery to the data recipient is identical.

Providing a web service

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

Calling the web service

The Mobilithek provides a web service client, defined based on [DATEXIIv3Push] defined web service client for calling the data receiver web services. Via the Mobilithek administration component, the data subscriber must store his service endpoint in the subscription configuration.

The broker system identifies these data recipient systems and starts a corresponding web service call.

If the transfer of the data could be completed successfully, the broker system expects a corresponding confirmation message from the data receiver system:

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

Synchronisation

A data receiver system can cause the Mobilithek to resend data packets. This can be particularly useful, in the context of publications that support delta deliveries, to synchronise to the current state of the Mobilithek data package buffer after a data receiver system start and not have to wait for the next delivery of a complete data package.

If the Mobilithek receives a confirmation message with the return status "**snapshotSynchronisationRequest**" in response to a data transmission, the Mobilithek will retransmit all data packets contained in the data packet buffer to the data receiver system in the order in which they were received. The following shows a confirmation message that initiates a synchronisation.

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

7.3 HTTPS interface

7.3.1 Data provider side

7.3.1.1 Client Pull HTTPS

In the client pull exchange procedure, the broker system of the Mobilithek cyclically requests the data provider system to deliver its data to the Mobilithek. The time interval used must be configured when configuring the data offering in the metadata directory. For this exchange, the rules of the snapshot pull from the [DATEXIIv3Annex] *Appendix C - "Snapshot Pull with simple http server" profile definition*.

It should be noted that the other 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 procedure which is mandatory for the Mobilithek. See also Appendix B - DATEX II HTTP Protocol Support.

Request to the data provider

The Mobilithek broker system sends an HTTPS GET request to the data provider system from which the data is to be fetched. The Mobilithek is able to identify data provider systems that have subscribed to a pull procedure and send requests to them at defined intervals.

The data provider must store the publication-specific server URL in the publication configuration via the administration component of the Mobilithek. The URL must be stored in full by the data provider. The Mobilithek does not add parameters to it, such as the publication ID.

Please also note the instructions in chapter 4.8 “Use of the "If-Modified-Since" header field”.

Response to the Mobilithek

After receiving the request, the data provider system must generate an HTTPS response whose message body consists of the requested DATEX II v3 data. A MessageContainer object is expected here that meets the minimum profile of the MessageContainer.xsd [DATEXIIv3Exc] is met. According to [DATEXIIv3Annex] *Appendix C - "Snapshot Pull with simple http server" profile definition, Basic request / response pattern*, the response must be in the content type "text/xml; charset=utf-8" and can be delivered GZIP-encoded.

The Mobilithek's broker system accepts this data and stores it in a packet buffer. If the delivered data package is a complete package, all data packages in the package buffer are replaced. A delta packet is appended to the list of data packets in the packet buffer.

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 <URL of schema
file>'
            modelBaseVersion='3'>
 <con:payload lang='en'
         xsi:type='sit:SituationPublication'
         modelBaseVersion='3'>
    ...
  </con:payload>
  <con:exchangeInformation modelBaseVersion='3'>
    <ex:exchangeContext>
      <ex:codedExchangeProtocol>snapshotPull</ex:codedExchangeProtocol>
      <ex:exchangeSpecificationVersion>3.0</ex:exchangeSpecificationVersion>
      <ex:supplierOrCisRequester/>
    </ex:exchangeContext>
    <ex:dynamicInformation>
      <ex:exchangeStatus>online</ex:exchangeStatus>
      <ex:messageGenerationTimestamp>2021-07-21T13:00:00
      </ex:messageGenerationTimestamp>
    </ex:dynamicInformation>
  </con:exchangeInformation>
</con:messageContainer>
```

7.3.1.2 Publisher Push HTTPS

The data provider system must send a data packet for a publication to the Mobilithek broker system.

Request to the Mobilithek broker system

The data provider system must send an HTTPS POST request with a message in the HTTP request body to the Mobilithek broker system.

The publication ID must be specified as a path element in the URL. The user data is passed in the HTTP request body.

The Content-Type header to be sent depends on the syntax to which the corresponding data offer is set.

It is the responsibility of the data provider to define the mandatory <exchangeInformation> element with its <exchangeContext> and <dynamicInformation> elements if it is a delta delivery. In order to provide the data in a standard compliant way, it should be noted that the <codedExchangeProtocol> element must be set to the value.

- "snapshotPull" for a complete data packet
- "deltaPull" for a delta data packet

should be set. This requires the data packet to be in a valid XML format (or JSON format if the data offer has been set with the JSON syntax). If the data offer does not specify a delta delivery, it will not check for the existence or content of the <codedExchangeProtocol> element.

Regardless, Mobilithek replaces the <codedExchangeProtocol> element with the value appropriate to the delivery protocol in each case (see Chapter 4.5.2) to allow standards-compliant data taker systems to process it correctly. Following the hint in chapter 4, the broker system only performs necessary validations of the response.

The URL of the broker system is structured as follows:

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

Response to the Data provider

The data provider system receives an HTTPS response to the request. The message body is empty; the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings in Table 25 apply.

Description	
Request	Request POST /mobilithek/api/v1.0/publication/datexv3/<publicationID>/snapShotPushService HTTP/1.1 Host: mobilithek.info Content-Type: text/xml oder application/xml or application/json Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Statuscodes	Standard HTTP1.1 Statuscodes [HTTP/1.1] The following Statuscodes have a special meaning: ▪ 400: The specified publication parameter is not numeric or the request is not structured correctly. ▪ 404: Publication parameter could not be assigned, the publication is no longer valid, or no value was specified after the slash in the URL path. ▪ 403: The user is not authorized to submit data via this endpoint for the specified publication or the publication has not been configured for delivery via HTTPS. ▪ 422: For an offering that supports delta delivery, no valid value was passed for the codedExchangeProtocol element or the syntax of the message passed in invalid.

Table 17: Request/Response between Provider System/Mobilithek during Publisher Push DatexIIv3 HTTPS

7.3.2 Data recipient side

7.3.2.1 Client Pull HTTPS

In the client pull exchange procedure, the data receiving system must request the mobile library to transmit the data.

Request to the Mobilithek

The data receiver system is to send an HTTPS GET request to the Mobilithek. Based on the subscription ID, the associated packet buffer as well as the data packet is determined. Alternatively, an HTTPS POST request can be used.

The URL of the broker system for XML based publications is structured as follows:

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

The URL of the broker system for JSON based publications is structured as follows:

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

Please also note the instructions in chapter 4.8 “Use of the "If-Modified-Since" header field”.

Response to the data recipient

The broker system of the Mobilithek generates an HTTPS response after receiving the request. For this purpose, the corresponding packet buffer and the appropriate data packet are determined based on the subscription ID. The content of the data packet is transmitted to the data recipient in the body of the response. According to the 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. In addition, the response has the content type "text/xml; charset=utf-8" resp. "application/json; charset=utf-8" and is - in deviation from the standard - exclusively sent in GZIP compressed form. **Requests with the "identity encoding" or other compression formats are acknowledged with the error code HTTP 406 (Not Acceptable)**.

Status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the Table 18 described meanings apply:

Description	
Request	Request GET /mobilithek/api/v1.0/subscription/datexv3?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <messageContainer > ... </messageContainer>
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 204: No data packet in the packet buffer for subscription ▪ 304: No data packet in the packet buffer that is younger than the timestamp in the "if-modified-since" header. ▪ 400: No subscription parameter specified in the request or missing Accept-Encoding header. ▪ 403: The user is not authorised to retrieve this subscription via this endpoint or it is not a DATEX2 V3 publication associated with this subscription. ▪ 404: Subscription parameter is not numeric or subscription is no longer valid or does not exist. ▪ 406: gzip not specified in the "Accept-Encoding" request header. ▪ 503: Service Unavailable (e.g. during maintenance)

Table 18 : Request/Response between Mobilithek/Data Recipient System on Client Pull HTTPS

8 Container

8.1 SOAP interface

8.1.1 Data provider side

8.1.1.1 Client Pull SOAP

In the client pull SOAP exchange procedure, the broker system of the Mobilithek cyclically requests the data provider system to deliver its data to the Mobilithek. The time interval used must be configured when configuring the data offering in the metadata directory.

Providing a web service

The data provider system must offer a web service with the method pullContainerDataBroker, which expects as input the parameters publication ID (type publicationId) and an optional timestamp (type timestamp) with a creation date according to the elements of the container model schema.

Via the administration component of the Mobilithek, the data provider must store the service endpoint in the URL attribute of the publication configuration. The Mobilithek broker system assumes that the data packets submitted by the data provider system for the specified URL belong to the configured publication, regardless of the publication ID in the SOAP request.

The data provider system must generate and return a data package (type containerdata) in container format. Following the hint in chapter 4 the Mobilithek broker system accepts all data packets, provided they have a valid XML structure.

Important note

The administration component of the Mobilithek makes it possible to test the accessibility of the specified URL of the data provider. Since this is possible at a time when the publication ID is not yet known in the Mobilithek, the Mobilithek will execute this test request with a random ID. The data provider system must therefore generate a valid data packet regardless of the publication ID contained in the request.

Calling a web service

The broker system of the Mobilithek provides a web service client defined according to the container format specification [MCS] for calling web services.

The broker system identifies the data provider systems that have subscribed to a pull procedure and the associated service endpoint in the metadata directory and calls them cyclically according to the configured publication frequency. The data received after the call are temporarily stored in corresponding packet buffers for delivery to potential data receivers. Any previous data package that may still exist is replaced in the process.

8.1.1.2 Publisher Push SOAP (Container)

With the Publisher Push exchange procedure, the data provider system must deliver the data to the Mobilithek on its own. A corresponding SOAP interface must be used. Whether the data is generated because of an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant for the functioning of the Mobilithek. The exchange mechanism is identical in both cases.

Providing a web service

The broker system of the Mobilithek offers a web service with the method pushContainerData, which expects the data structure of the container format filled with the publication ID in the header element and a data package in the body element as input 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'?>
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns3:container xmlns="https://www.w3.org/2000/09/xmldsig#"
                   xmlns:ns2="https://schemas.xmlsoap.org/ws/2002/07/utility"
                   xmlns:ns3="https://ws.bast.de/container/TrafficDataService">
      <ns3:header>
        <ns3:Identifier>
          <ns3:publicationId>12345</ns3:publicationId>
        </ns3:Identifier>
      </ns3:header>
      <ns3:body>
        <ns3:binary id="test-id-bin" type="hexBinary">dGVzdC10ZXh0&#xD;.</ns3:binary>
        <ns3:xmlschema="test-schema" id="test-id-xml"/>
      </ns3:body>
    </ns3:container>
  </soapenv:Body>
</soapenv:Envelope>
```

Calling the web service

The data provider system must provide a web service client according to the container format specification [MCS] to call the web service.

The SOAP endpoint of the broker system is:

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

Error codes

HTTP Response Code	SOAP Response	Description
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalid XML or invalid SOAP request
	SOAP Fault faultcode: Client faultstring: SOAP action cannot be determined	Multiple SOAP methods identified in the request, or the first element in the SOAP body is not the supported SOAP method: <container>.
	SOAP Fault faultcode: Client faultstring: numeric publicationID expected, found non-numeric publicationId	The specified publication ID is not numeric
	SOAP Fault faultcode: Client faultstring: tag <publicationID> expected according to WDSL, element not found	<publicationID> Element not found in the request.
	SOAP Fault faultCode: Client faultString = publicationId for this organisation not found	The user assigned to the machine account is not authorised to provide data for the specified publication, or the specified publication does not exist, or The HTTPS SOAP delivery protocol was not selected for the specified publication.

Table 19: Error codes for SOAP Container Publisher Push

8.1.2 Data recipient side

8.1.2.1 Client Pull SOAP

In the client pull SOAP exchange method, the data receiver system must request the mobile library to send data to the data receiver system.

Providing a web service

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

Example:

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

Calling the web service

The data receiving system must provide a web service client according to the container format specification [MCS] for calling the web service.

The SOAP endpoint of the broker system is:

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

Error codes

HTTP Response Code	SOAP Response	Description
500	SOAP Fault faultcode: Client faultstring: invalid - XML	Invalid XML or invalid SOAP request
	SOAP Fault faultcode: Client	Multiple SOAP methods identified in the request, or

HTTP Response Code	SOAP Response	Description
	faultstring: SOAP action cannot be determined	the first element in the SOAP body is not the supported SOAP method: <pullContainerDataClientRequestEl>.
	SOAP Fault faultcode: Client faultstring: Expected to find numerical value in subscriptionId, found non-numeric	The specified subscription ID is not numeric
	SOAP Fault faultcode: Client faultstring: Expected to find element subscriptionId in request, but did not find	<subscriptionID> element not found in the request.
	SOAP Fault faultCode: Client faultString: Accessible subscriptionId for organisation not found	The user assigned to the machine account is not authorised to obtain data for the specified subscription, or the specified subscription does not exist, or the specified subscription is not a container publication
	SOAP Fault faultCode: Server faultString: No data package available	No data packet available in the packet buffer
400	empty body	Request faulty, e.g. "Accept-Encoding" header missing
406	empty body	gzip not specified in the "Accept-Encoding" request header.

Table 20: Error codes for SOAP Container Consumer Pull

8.1.2.2 Publisher Push SOAP

In the Publisher Push exchange procedure, the Mobilithek delivers the data to the data recipient systems on its own. A corresponding SOAP interface is used for this. Whether the data is generated because of an event (on occurrence) or periodically (periodic) and delivered to the Mobilithek is irrelevant, the mechanism for delivery to the data recipient is identical.

Providing a web service

The data receiver system must offer a web service with the method pushContainerData, which is defined based on the container format specification [MCS]. A data packet of the container format type (type containerdata) must be accepted as input and a status message (also of type containerdata) must be delivered as output.

Calling the web service

The broker system of the Mobilithek provides a web service client defined based on the container format specification [MCS] for calling up the data recipient web services. Via the administration component of the Mobilithek, the data recipient must store his service endpoint in the URL attribute of the subscription configuration.

The broker system identifies these data recipient systems and starts a corresponding web service call.

If the transfer of the data could be completed successfully, the broker system expects a corresponding status message from the data receiver system.

If the data receiving system does not acknowledge the transmission as successful, the transmission is repeated according to chapter 4.7.2 the transmission is repeated.

If the data receiving system is technically not accessible, HTTP requests are sent to the service end point extended by the suffix "? wsdl". Only when this HTTP request is acknowledged with a status of 200 will the data recipient system be supplied again.

8.2 HTTPS interface

8.2.1 Data provider side

8.2.1.1 Client Pull HTTPS

The Mobilithek broker system cyclically requests the data provider system to deliver a data package for a publication to the Mobilithek. The time interval used must be configured when configuring the data offering in the metadata directory.

Request to the data provider

The broker system sends an HTTPS GET request to the data provider system. The broker system uses the URL stored in the publication description as URL.

Example:

```
GET <URL according to publication description>
content-type: text/plain
accept-encoding: gzip
```

The URL must be stored in full by the data provider. Mobilithek does not add parameters to this, such as the publication ID.

Response to Mobilithek

The data provider system must respond to the request with an HTTPS response. The content type of the response must be "text/xml" and should be available as GZIP encoding. The Mobilithek can also process non-compressed content. The message body must consist of the requested data package.

The payload transmitted in the response body is only stored in the broker system of the Mobilithek if the data provider system responds with a status 200.

See also the notes in Chapter 4.8.1 on using the If-Modified-Since and Last-Modified HTTP headers.

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

8.2.1.2 Publisher Push HTTPS

The data provider system must send a data package for a publication to the broker system of the mobile library.

Request to the broker system of the Mobilithek

The data provider system must send an HTTPS POST request with a message in container format to the broker system of the mobile library. The publication ID must be passed in the header element and the user data in the body element of the container message.

The URL of the broker system is structured as follows:

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

Example:

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

Response to the data provider

The data provider system receives an HTTPS response to the request. The message body is empty, the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings described in Table 21 apply.

Description	
Request	Request POST /mobilithek/api/v1.0/publication/container HTTP/1.1 Host: mobilithek.info Content-Type : application/xml Accept-Encoding: gzip <container> ... </container>
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 400: No publication parameter, the publication parameter is not numeric or the request is not structured correctly, e.g. it does not contain valid XML. ▪ 403: The user is not authorised to deliver data via this endpoint for the specified publication or the publication has not been specified for HTTPS container delivery. ▪ 404: Publication parameter could not be assigned to a valid publication

Table 21 : Request/Response between Data Provider System/Mobile Library at Publisher Push HTTPS

8.2.2 Data recipient side

8.2.2.1 Client Pull HTTPS

In the client pull exchange procedure, the data receiver system must request the broker system of the mobile library to transmit the data. Which subscription is involved must be specified by a request parameter.

Request to the Mobilithek

The data receiver system must send an HTTPS GET request to the Mobilithek. The subscription ID to which a data packet is to be delivered must be transmitted as a parameter.

The URL of the broker system is structured as follows:

```
https://mobilithek.info:8443/mobilithek/api/v1.0/container/subscription?subscriptionID=<Sub
scription-ID>
```

See also the notes in Chapter 4.8 “Use of the "If-Modified-Since" header field in the HTTPS protocol”.

Response to the data receiver system

The broker system of the Mobilithek generates an HTTPS response after receiving the request. The status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings described in Table 22 apply as status codes. The content type of the response is "text/xml" and is sent in GZIP compressed form. The message body of the response consists of the requested data package.

Description	
Request	Request GET /mobilithek/api/v1.0/container/subscription?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK Content-Type: text/xml Content-Length: xx <container> ... </container>
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 204: No data packet in the packet buffer for subscription. ▪ 304: No data packet in the packet buffer that is younger than the timestamp in the "if-modified-since" header. ▪ 400: No subscription parameter or missing Accept-Encoding Header ▪ 403: The user is not authorised to retrieve this subscription via this endpoint or the associated publication is not provided in container format. ▪ 404: No or no longer valid subscription found for the subscription parameter. ▪ 405: The parameter ? subscriptionId= was not specified or no value was specified for the parameter. ▪ 406: gzip not specified in the "Accept-Encoding" request header.

Table 22 : Request/Response between Mobilithek/Data Recipient System at Client Pull HTTPS

8.2.2.2 Publisher Push HTTPS

The Mobilithek broker system sends a data packet for a subscription to a data receiver system.

Request to the data receiver system

The broker system of the mobile library sends an HTTPS POST request to the data recipient system, in which the subscription ID in the header element and the user data in the body element of the container message are passed.

Via the administration component of the Mobilithek, the data recipient must store his URL in the subscription configuration.

The URL must be stored in full by the data recipient. Mobilithek does not add parameters to it, such as the subscription ID.

Response to the Mobilithek

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

The message body should be empty, the status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings described in Table 23 apply as status codes.

Description	
Request	Request POST /data delivery HTTP/1.1 Host: datennehmerhost Content-Type : text/xml Accept-Encoding: gzip <container> ... </container>
Response	Response HTTP/1.1 200 OK
Status codes	Status Codes Standard HTTP1.1 Status Codes [HTTP/1.1]

Table 23 : Request/response between broker system of the mobile library/data receiver system at the publisher Push HTTPS

With reference to the explanations in chapter 4.7.2 status codes other than 200 will result in a repeated attempt to transmit the data packet.

If the data receiving system is technically not accessible, HTTP head requests are sent to the stored URL. Only when the HTTP head request is acknowledged by the data receiving system with a status of 200 will it be delivered again.

9 Other data formats

The Mobilithek supports the delivery of any data format without the need to pack it into the container format.

9.1 Data provider side

9.1.1 Client Pull HTTPS

The Mobilithek broker system cyclically requests the data provider system to deliver a data package for a publication to the Mobilithek. The time interval used must be configured when configuring the data offering in the metadata directory.

9.1.1.1 Request to the data provider

The broker system sends an HTTPS GET request to the data provider system. The broker system uses the URL stored in the publication description as URL.

Example:

```
GET <URL according to publication description>
content-type: text/plain
accept-encoding: gzip
```

The URL must be stored in full by the data provider. The Mobilithek does not add parameters such as the publication ID.

9.1.1.2 Response to the Mobilithek

The data provider system must respond to the request with an HTTPS response. The content type of the response should correspond to the type of the transmitted payload and should be available as GZIP encoding. Non-compressed content can also be processed by the Mobilithek. Mobilithek does not carry out any checks regarding the consistency of the specified Content-Type and the transmitted payload.

The message body must consist of the requested data package. The standard HTTP status codes [HTTP/1.1] are to be used as status codes. The payload transmitted in the response body is only stored in the broker system of the Mobilithek if the data provider system responds with a status 200.

See also the notes in Chapter 4.8.1 on using the If-Modified-Since and Last-Modified HTTP headers.

Description	
Request	GET <URL according to publication description> HTTP/1.1 Host: Data provider host Accept-Encoding: gzip
Response	HTTP/1.1 200 OK Content-Type: text/csv Content-Length: xx [Data package]
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1]

Table 24: Request/response between data provider system/mobile library with client pull HTTPS

9.1.2 Publisher Push HTTPS

The data provider system must send a data package for a publication to the broker system of the mobile library.

9.1.2.1 Request to the Mobilithek broker system

The data provider system must send an HTTPS POST request with a message in the HTTP request body to the broker system of the mobile library.

The publication ID must be specified as a path element in the URL. The user data is transferred in the HTTP request body

The URL of the broker system is structured as follows:

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

9.1.2.2 Response to the data provider

The data provider system receives an HTTPS response to the request. The message body is empty, the standard HTTP status codes can occur as status codes [HTTP/1.1] can occur, whereby the meanings in Table 25 apply.

Description	
Request	Request POST /mobilithek/api/v1.0/publication/<publicationID> HTTP/1.1 Host: mobilithek.info Content-Type: text/csv Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 400: The specified publication parameter is not numeric or the request is not structured correctly. ▪ 404: Publication parameter could not be assigned, the publication is no longer valid or no value was specified after the slash in the URL path. ▪ 403: The user is not authorised to submit data via this endpoint for the specified publication or the publication has not been configured for delivery via HTTPS.

Table 25: Request/response between data provider system/mobile library at publisher push HTTPS

9.2 Data recipient side

9.2.1 Client Pull HTTPS

In the client pull exchange procedure, the data receiver system must request the broker system of the mobile library to transmit the data. Which subscription is involved must be specified by a request parameter.

Request to the Mobilithek

The data receiver system must send an HTTPS GET request to the Mobilithek. The subscription ID to which a data packet is to be delivered must be transmitted as a parameter.

The URL of the broker system is structured as follows:

```
https://mobilithek.info:8443/mobilithek/api/V1.0/subscription?
subscriptionID=<subscriptionId>
```

See also the notes in Chapter 4.8 “Use of the "If-Modified-Since" header field in the HTTPS protocol”.

Response to the data receiver system

The broker system of the Mobilithek generates an HTTPS response after receiving the request. The status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings described in following Table 26 apply. The content type of the response is of the same type as delivered by the data provider (in the following table text/csv is assumed) and is sent GZIP-compressed. The message body of the response consists of the requested data package.

Description	
Request	Request GET /mobilithek/api/V1.0/subscription?subscriptionID=2000000 HTTP/1.1 Host: mobilithek.info Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK Content-Type: text/csv Content-Length: xx column1, column2, further content
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1] The following status codes have a special meaning: ▪ 204: No data packet in the packet buffer for subscription. ▪ 304: No data packet in the packet buffer that is younger than the timestamp in the "if-modified-since" header. ▪ 400: No subscription parameter, or numeric subscription parameter, or missing Accept-Encoding header. ▪ 403: The user is not authorised to retrieve this subscription via this endpoint or the publication cannot be retrieved via this endpoint. ▪ 404: No or no longer valid subscription found for the subscription parameter. ▪ 405: The parameter ? subscriptionId= was not specified or no value was specified for the parameter. ▪ 406: gzip not specified in the "Accept-Encoding" request header.

Table 26: Request/Response between Mobilithek/Data Recipient System on Client Pull HTTPS

9.2.2 Publisher Push HTTPS

The Mobilithek broker system sends a data packet for a subscription to a data receiver system.

Request to the data receiver system

The broker system of the mobile library sends an HTTPS POST request to the data receiver system, in which the subscription ID in the header element and the user data in the HTTP request body are transferred. The content type of the request is of the same type as specified in the submission by the data provider (in the following table text/csv is assumed).

Via the administration component of the Mobilithek, the data recipient must store his URL in the subscription configuration.

The URL must be stored in full by the data recipient. Mobilithek does not add parameters to it, such as the subscription ID.

Response to the Mobilithek

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

The message body should be empty, the status codes can be the standard HTTP status codes [HTTP/1.1] can occur as status codes, whereby the meanings described in the following Table 27 apply as status codes.

Description	
Request	Request POST /data delivery HTTP/1.1 Host: datennehmerhost Content-Type : text/csv Accept-Encoding: gzip
Response	Response HTTP/1.1 200 OK
Status codes	Standard HTTP1.1 Status Codes [HTTP/1.1]

Table 27: Request/response between broker system of the Mobilithek/data receiver system at the publisher Push HTTPS

With reference to the explanations in chapter 4.7.2 status codes other than 200 will result in a repeated attempt to transmit the data packet.

If the data receiving system is technically not accessible, HTTP head requests are sent to the stored URL. Only when the HTTP head request is acknowledged by the data receiving system with a status of 200 will it be delivered again.

Identity encoding is not supported by Mobilithek.

10 Certificate-based M2M communication

The security component of the Mobilithek requires a certificate-based data exchange between the data provider system and the platform on the one hand and between the platform and the data recipient system on the other.

This chapter first provides an overview of the functions of the security component and then describes the steps that data providers and data recipients must take to request certificates and set them up for M2M communication.

The certificate is applied for at the administration component of the Mobilithek by the organisation's administrator and is created after the application and sent to the organisation's administrator by e-mail. The password required for the signature is sent by SMS to the stored mobile number.

Finally, the data provider system/data recipient system must integrate the certificate into their IT infrastructure so that the data exchange with the Mobilithek can be authenticated.

10.1 Tasks of the security component

The security component is responsible for implementing the security aspects of the Mobilithek. This includes in particular the authentication of data provider systems and data recipient systems that want to communicate with the Mobilithek.

Before the data packets arriving at the Mobilithek are accepted, their origin must be verified. This includes the authentication of the data provider system belonging to the data package by means of a digital certificate. Each data provider system must have a valid certificate with which it logs on to the platform. The security component authenticates the certificate sent by the data provider system within the Mobilithek.

Before a data packet is sent to a data receiver system, the identity of the data receiver system must be verified. Each data recipient system must authenticate itself to the Mobilithek by means of a digital certificate. The security component authenticates the certificate sent by the data recipient sys tem within the Mobilithek.

The confidentiality of the communication between the data provider system and the Mobilithek on the one hand and the Mobilithek and the data recipient system on the other hand is guaranteed by the exclusive use of SSL/TLS transport encryption.

The security component requires standard-compliant [X.509v3]-certificates for authentication; see also [PKI]. The certificates must be technically integrated into the HTTPS connection to the data recipient and data provider systems via a client-side, certificate-based connection setup. The presented certificates are checked for validity and against a revocation list.

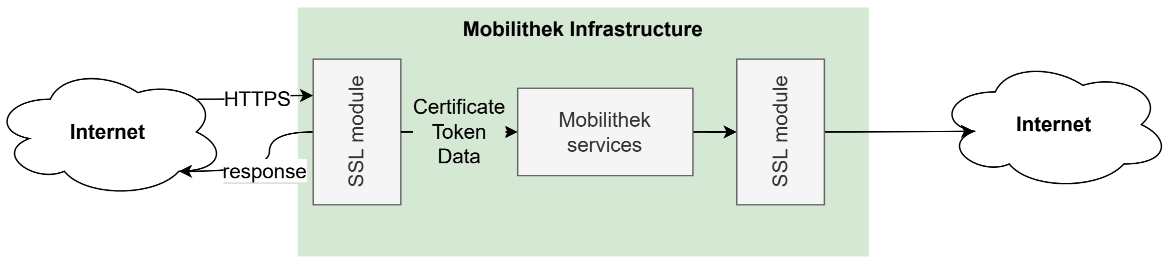


Figure 4: Overview of the security architecture

The SSL module in Figure 4 sends a certificate request to the sender for specified URLs and checks the certificate received for validity and against a revocation list. It then forwards the certificate to the security component of the Mobilithek.

10.2 Request machine certificate

The Mobilithek operator mediates between the data provider and data recipient system. Data providers and data recipients apply for one or more machine certificates via the Mobilithek administration GUI as part of their registration. However, the certificate is then sent to them by the Mobilithek.

To request a machine certificate, you must already be registered with your organisation on Mobilithek.

How to apply for a machine certificate via the Mobilithek is described in [FAQ] described.

10.3 Install machine certificate and exhibitor certificate

In the Apache web server, include the machine certificate as follows:

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

Enter the corresponding private key as follows:

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

In addition, you must store the issuer certificate in the web server:

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

The certificate is encrypted via the key with the password that was sent to you via SMS. Use the password to decrypt.

Further explanations of these directives can be found in the mod_ssl documentation:

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

Note: If you receive the machine certificate and the issuer certificate within a common p12 file, you must extract both certificates from this file and then install them. The instructions for this can be found in chapter 11.

10.4 Authentication of the Mobilithek as a web client

If the Mobilithek functions as a web client in M2M communication, it authenticates itself with its server certificate, provided the web server on the data provider or data recipient side has activated this option. Data provider and data recipient systems should activate this option and verify the certificate in order to determine that the requests were actually made by the Mobilithek.

The certificate chain required for verification can be downloaded from the download area (https://mobilithek.info/help/download) of the Mobilithek in the category Certificates and must be stored in the data provider or data receiver system.

In order to establish an SSL connection, it is necessary that a valid server certificate or a machine certificate issued by the Mobilithek is installed on the server of the data provider or data recipient side.

Note: Do not use the Mobilithek server certificate for verification. This is replaced on a regular basis.

10.5 Authentication of data provider/data recipient web clients

If the data provider or data recipient system functions as a web client in M2M communication, it must authenticate itself to the Mobilithek with its machine certificate. The platform only accepts requests from systems that are registered in the metadata directory. Based on the certificate, the machine can be assigned to the organisation. Furthermore, it can be checked whether the organisation is the owner of the publication or subscription for which data exchange is to take place.

The server certificate used by Mobilithek is signed by Mobilithek. It is therefore necessary to install it in the system's "Truststore". The certificate can be downloaded from the download area (https://mobilithek.info/help/download) of the Mobilithek in the category Certificates.

11 Appendix A - prepare p12 file for Apache server configuration

The Apache server configuration cannot process files of type p12. Manual steps are required for the preparation, which are described in the following chapter:

First export the keys and certificates. Execute the following command in the command line:

```
openssl.exe pkcs12 -in <p12-file> -out <collect-file.pem>
```

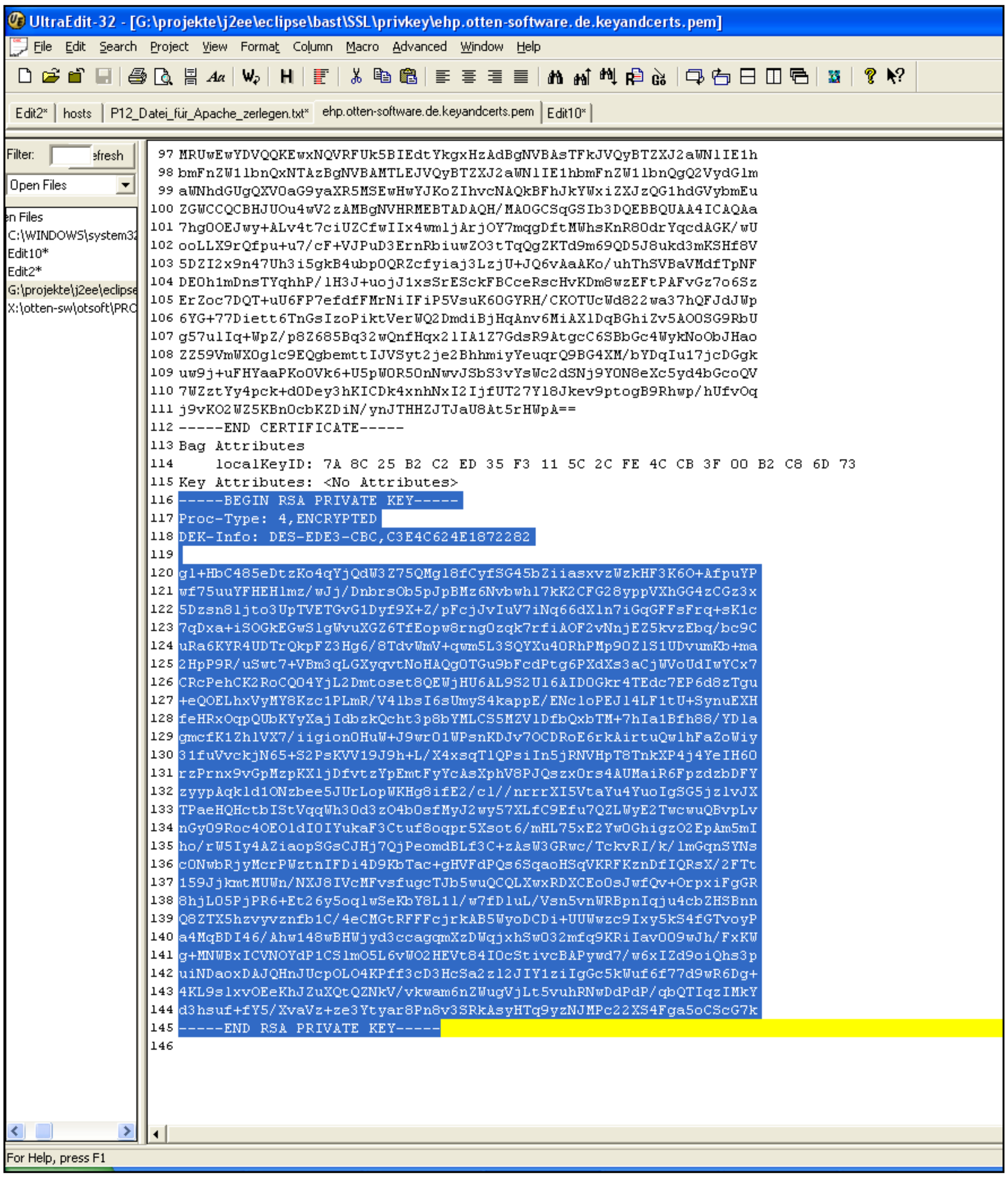
Example:

```
openssl.exe pkcs12 -in ehp.example.com.p12 -out ehp.example.com.keyandcerts.pem
```

Enter the certificate passwords in the Openssl environment:

```
>Enter Import Password:      <password from SMS>
>MAC verified OK
>Enter PEM pass phrase:      <self-assigned pass phrase for the key>
>Verifying - Enter PEM pass phrase: <repeat the self-assigned pass phrase for the key>.
```

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


UltraEdit-32 - [G:\projekte\j2ee\eclipse\bast\SSL\privkey\ehp.otten-software.de.keyandcerts.pem]
File Edit Search Project View Format Column Macro Advanced Window Help
Edit2* hosts P12_Datei_für_Apache_zerlegen.txt* ehp.otten-software.de.keyandcerts.pem Edit10*
Filter: efresh
Open Files
en Files
C:\WINDOWS\system32
Edit10*
Edit2*
G:\projekte\j2ee\eclipse
X:\otten-sw\otsoft\PRO
For Help, press F1

```
MRUwEwYDVQQKEwxNQVRFUk5BIEdtYkgxHzAdBgNVBAsTFkJVQyBTZXJ2aWNlIE1h
bmFnZW1lbnQxNTAzBgNVBAMTLEJVQyBTZXJ2aWNlIE1hbmFnZW1lbnQgQ2VydGlm
aWNhdGUgQXV0aG9yaXR5MSEwHwYJKoZIhvcNAQkBFhJkYWxiZXJzQG1hdGVybmEu
ZGWCCQCBHJUOu4wV2zAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4ICAQAa
7hgOOEJwy+ALv4t7ciUZCfwIIx4wmljArjOY7mqgDftMWhsKnR8OdrYqcdAGK/wU
ooLLX9rQfpu+u7/cF+VJPuD3ErnRbiuwZO3tTqQgZKTd9m69QD5J8ukd3mKSHf8V
5DZI2x9n47Uh3i5gkB4ubpOQRZcfyiaj3LzjU+JQ6vAaAKo/uhThSVBaVMdfTpNF
DEOh1mDnsTYqhhP/lH3J+uojJ1xsSrESckFBCceRscHvKDm8wzEFtPAFvGz7o6Sz
ErZoc7DQT+uU6FP7efdfFMrNiIFiP5VsuK6OGYRH/CKOTUcWd822wa37hQFJdJWp
6YG+77Diett6TnGsIzoPiktVerWQ2DmdiBjHqAnv6MiAX1DqBGhiZv5AOOSG9RbU
g57ulIq+WpZ/p8Z685Bq32wQnfHqx2lIA1Z7GdsR9AtgcC6SBbGc4WykNoObJHao
ZZ59VmWXOglc9EQgbemttIJVSyt2je2BhhmiyYeuqrQ9BG4XM/bYDqIu17jcDGgk
uw9j+uFHYaaPKoOVk6+U5pWOR5OnNwvJSbS3vYsWc2dSNj9YON8eXc5yd4bGcoQV
7WZztYy4pck+dODey3hKICDk4xnhNxI2IjfUT27Yl8Jkev9ptogB9Rhwp/hUfvOq
j9vKO2WZ5KBnOcbKZDiN/ynJTHHZJTJaU8At5rHWpA==
-----END CERTIFICATE-----
Bag Attributes
    localKeyID: 7A 8C 25 B2 C2 ED 35 F3 11 5C 2C FE 4C CB 3F 00 B2 C8 6D 73
Key Attributes: <No Attributes>
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,C3E4C624E1872282

gl+HbC485eDtzKo4qYjQdW3Z75QMgl8fCyfSG45bZiiasxvzWzkHF3K6O+AfpuYP
wf75uuYFHEHlmz/wJj/DnbrsOb5pJpBMz6Nvbwhl7kK2CFG28yppVXhGG4zCGz3x
5Dzsn8ljto3UpTVETGvG1Dyf9X+Z/pFcjJvIuV7iNq66dXln7iGqGFFsFrq+sK1c
7qDxa+iSOGkEGwSlgWvuXGZ6TfEopw8rngOzqk7rfiAOF2vNnjEZ5kvzEbq/bc9C
uRa6KYR4UDTrQkpFZ3Hg6/8TdvWmV+qwm5L3SQYXu4ORhPMp9OZlS1UDvumKb+ma
2HpP9R/uSwt7+VBm3qLGXyqvtNoHAQgOTGu9bFcdPtg6PXdXs3aCjWVoUdIwYCx7
CRcPehCK2RoCQO4YjL2Dmtoset8QEWjHU6AL9S2Ul6AIDOGkr4TEdc7EP6d8zTgu
+eQOELhxVyMY8Kzc1PLmR/V4lbsI6sUmyS4kappE/ENcloPEJl4LF1tU+SynuEXH
feHRxOqpQUbKYyXajIdbzkQcht3p8bYMLCS5MZVlDfbQxbTM+7hIa1Bfh88/YDla
gmcfK1ZhlVX7/iigionOHuW+J9wrO1WPsnKDJv7OCDRoE6rkAirtuQwlhFaZoWiy
31fuVvckjN65+S2PsKVV19J9h+L/X4xsqTlQPsiIn5jRNVHpT8TnkXP4j4YeIH6O
rzPrnx9vGpMzpKXljDfvtzYpEmtFyYcAsXphV8PJQszxOrs4AUMaiR6FpzdzbDFY
zyypAqkld1ONzbee5JUrLopWKHg8ifE2/cl//nrrrXI5VtaYu4YuoIgSG5jzlvJX
TPaeHQHctbIStVqqWh3Od3zO4bOsfMyJ2wy57XLfC9Efu7QZLWyE2TwcwuQBvpLv
nGyO9Roc4OEOldIOIYukaF3Ctuf8oqpr5Xsot6/mHL75xE2YwOGhigzO2EpAm5mI
ho/rW5Iy4AZiaopSGsCJHj7QjPeomdBLf3C+zAsW3GRwc/TckvRI/k/lmGqnSYNs
cONwbRjyMcrPWztnIFDi4D9KbTac+gHVFdPQs6SqaoHSqVKRFKznDfIQRsX/2FTt
159JjkmtMUWn/NXJ8IVcMFvsfugcTJb5wuQCQLXwxRDXCEoOsJwfQv+OrpxiFgGR
8hjLO5PjPR6+Et26y5oqlwSeKbY8L1l/w7fDluL/Vsn5vnWRBpnIqju4cbZHSBnn
Q8ZTX5hzvyvznfb1C/4eCMGtRFFFcjrkAB5WyoDCDi+UUWwzc9Ixy5kS4fGTvoyP
a4MqBDI46/Ahw148wBHWjyd3ccagqmXzDWqjxhSwO32mfq9KRiIavOO9wJh/FxKW
g+MNWBxICVNOYdP1CSlmO5L6vWO2HEVt84IOcStivcBAPywd7/w6xIZd9oiQhs3p
uiNDaoxDAJQHnJUcpOLO4KPff3cD3HcSa2zl2JIY1ziIgGc5kWuf6f77d9wR6Dg+
4KL9slxvOEeKhJZuXQtQZNkV/vkwam6nZWugVjLt5vuhRNwDdPdP/qbQTIqzIMkY
d3hsuf+fY5/XvaVz+ze3Ytyar8Pn8v3SRkAsyHTq9yzNJMPc22XS4Fga5oCScG7k
-----END RSA PRIVATE KEY-----
```

Figure 5: File <collectfile.pem>

Copy the part from

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

to

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

to a new file called `<server.key>`.

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

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

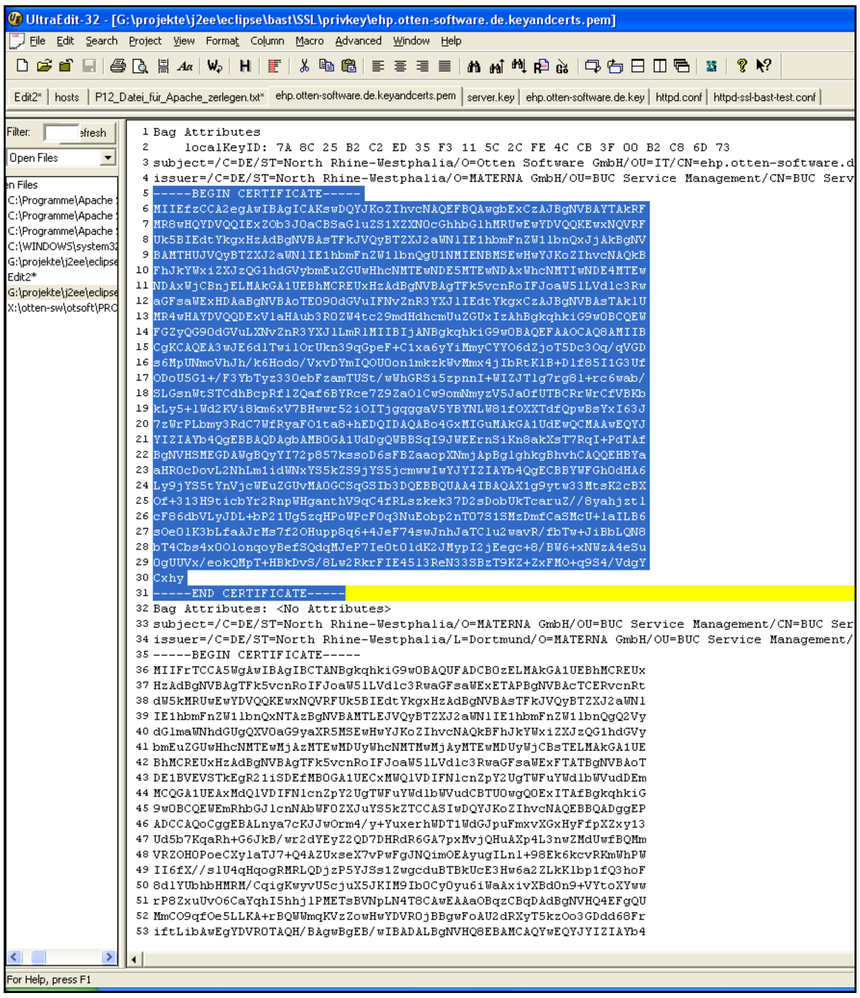
Example:

```
openssl rsa -in server.key -out ehp.example.com.key
> Enter pass phrase for server.key: <Enter the pass phrase you previously assigned
yourself>.
>writing RSA key
```

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

```
SSLCertificateKeyFile
```

The next step is to split the certificates into two files. First open the file `<collectionfile.pem>` with a text editor:



```
Bag Attributes
    localKeyID: 7A 8C 25 B2 C2 ED 35 F3 11 5C 2C FE 4C CB 3F 00 B2 C8 6D 73
subject=/C=DE/ST=North Rhine-Westphalia/O=Otten Software GmbH/OU=IT/CN=ehp.otten-software.d
issuer=/C=DE/ST=North Rhine-Westphalia/O=MATERNA GmbH/OU=BUC Service Management/CN=BUC Serv
-----BEGIN CERTIFICATE-----
MIIEfzCCA2egAwIBAgICAKswDQYJKoZIhvcNAQEFBQAwgbExCzAJBgNVBAYTAkRF
MR8wHQYDVQQIExZOb3J0aCBSaGluZS1XZXN0cGhhbGlhMRUwEwYDVQQKEwxNQVRF
Uk5BIEdtYkgxHzAdBgNVBAsTFkJVQyBTZXJ2aWNlIE1hbmFnZW1lbnQxJjAkBgNV
BAMTHUJVQyBTZXJ2aWNlIE1hbmFnZW1lbnQgU1NMIENBMSEwHwYJKoZIhvcNAQkB
FhJkYWxiZXJzQG1hdGVybmEuZGUwHhcNMTEwNDE5MTEwNDAxWhcNMTIwNDE4MTEw
NDAxWjCBnjELMAkGA1UEBhMCREUxHzAdBgNVBAgTFk5vcnRoIFJoaW5lLVdlc3Rw
aGFsaWExHDAaBgNVBAoTE09OdGVuIFNvZnR3YXJlIEdtYkgxCzAJBgNVBAsTAklU
MR4wHAYDVQQDExVlaHAub3ROZW4tc29mdHdhcmUuZGUxIzAhBgkqhkiG9w0BCQEW
FGZyQG90dGVuLXNvZnR3YXJlLmRlMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEA3wJE6dlTwilOrUkn39qGpeF+C1xa6yYiMmyCYYO6dZjoT5Dc3Oq/qVGD
s6MpUNmoVhJh/k6Hodo/VxvDYmIQOUOon1mkzkWvMmx4jIbRtKlB+Dlf85I1G3Uf
ODoU5G1+/F3YbTyz33OebFzamTUSt/wWhGRSi5zpnnI+WIZJTlg7rg8l+rc6wab/
SLGsnWtSTCdhBcpRflZQaf6BYRce7Z9ZaOlCw9omNmyzV5JaOfUTBCRrWrCfVBKb
kLy5+lWd2KVi8km6xV7BHwwr52iOITjgqggaV5YBYNLW81fOXXTdfQpwBsYxI63J
7zWrPLbmy3RdC7WfRyaFO1ta8+hEDQIDAQABo4GxMIGuMAkGA1UdEwQCMAAwEQYJ
YIZIAYb4QgEBBAQDAgbAMB0GA1UdDgQWBBSqI9JWEErnSiKn8akXsT7RqI+PdTAf
BgNVHSMEGDAWgBQyYI72p857kssoD6sFBZaaopXNmjApBglghkgBhvhCAQQEHBYa
aHR0cDovL2NhLm1idWNxYS5kZS9jYS5jcmwwIwYJYIZIAYb4QgECBBYWFGh0dHA6
Ly9jYS5tYnVjcWEuZGUvMA0GCSqGSIb3DQEBBQUAA4IBAQAX1g9ytw33MtsK2cBX
Of+313H9ticbYr2RnpWHganthV9qC4fRLszkek37D2sDobUkTcaruZ//8yahjztl
cF86dbVLyJDL+bP21Ug5zqHPoWPcFOq3NuEobp2nTO7S1SMzDmfCaSMcU+laILB6
sOe0lK3bLfaAJrMs7f2OHupp8q6+4JeF74swJnhJaTClu2wavR/fbTw+JiBbLQN8
bT4Cbs4xOOlonqoyBefSQdqMJeP7IeOtOldK2JMypI2jEegc+8/BW6+xNWzA4eSu
OgUUVx/eokQMpT+HBkDvS/8Lw2RkrFIE45l3ReN33SBzT9KZ+ZxFMO+q9S4/VdgY
Cxhy
-----END CERTIFICATE-----
Bag Attributes: <No Attributes>
subject=/C=DE/ST=North Rhine-Westphalia/O=MATERNA GmbH/OU=BUC Service Management/CN=BUC Ser
issuer=/C=DE/ST=North Rhine-Westphalia/L=Dortmund/O=MATERNA GmbH/OU=BUC Service Management/
-----BEGIN CERTIFICATE-----
MIIFrTCCA5WgAwIBAgIBCTANBgkqhkiG9w0BAQUFADCBOzELMAkGA1UEBhMCREUx
HzAdBgNVBAgTFk5vcnRoIFJoaW5lLVdlc3RwaGFsaWExETAPBgNVBAcTCERvcnRt
dW5kMRUwEwYDVQQKEwxNQVRFUk5BIEdtYkgxHzAdBgNVBAsTFkJVQyBTZXJ2aWNl
IE1hbmFnZW1lbnQxNTAzBgNVBAMTLEJVQyBTZXJ2aWNlIE1hbmFnZW1lbnQgQ2Vy
dGlmaWNhdGUgQXV0aG9yaXR5MSEwHwYJKoZIhvcNAQkBFhJkYWxiZXJzQG1hdGVy
bmEuZGUwHhcNMTEwMjAzMTEwMDUyWhcNMTMwMjAyMTEwMDUyWjCBsTELMAkGA1UE
BhMCREUxHzAdBgNVBAgTFk5vcnRoIFJoaW5lLVdlc3RwaGFsaWExFTATBgNVBAoT
DE1BVEVSTkEgR21iSDEfMB0GA1UECxMWQlVDIFNlcnZpY2UgTWFuYWdlbWVudDEm
MCQGA1UEAxMdQlVDIFNlcnZpY2UgTWFuYWdlbWVudCBTU0wgQ0ExITAfBgkqhkiG
9w0BCQEWEmRhbGJlcnNAbWF0ZXJuYS5kZTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBALnya7cKJJwOrm4/y+YuxerhWDT1WdGJpuFmxvXGxHyFfpXZxy13
Ud5b7KqaRh+G6JkB/wr2dYEyZ2QD7DHRdR6GA7pxMvjQHuAXp4L3nwZMdUwfBQMm
VRZOHOPoeCXylaTJ7+Q4AZUxseX7vPwFgJNQimOEAyugILnl+98Ek6kcvRKmWhPW
II6fX//slU4qHqogRMRLQDjzP5YJSs1ZwgcduBTBkUcE3Hw6a2ZLkKlbp1fQ3hoF
8dlYUbhbHMRM/CqigKwyvU5cjuX5JKIM9IbOCyOyu6iWaAxivXBdOn9+VYtoXYww
rP8ZxuUvO6CaYqhI5hhjlPMETsBVNpLN4T8CAwEAAaOBqzCBqDAdBgNVHQ4EFgQU
MmCO9qfOe5LLKA+rBQWWmqKVzZowHwYDVR0jBBgwFoAU2dRXyT5kzOo3GDdd68Fr
iftLibAwEgYDVR0TAQH/BAgwBgEB/wIBADALBgNVHQ8EBAMCAQYwEQYJYIZIAYb4
```

Figure 6: File <collectionfile.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 Mobilithek client certificate incl. 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.example.com.crt".
SSLCertificateKeyFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.key\ehp.example.com.key".
SSLCertificateChainFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.crt\bast_cert_chain.crt".
SSLCACertificateFile "C:\Programme\Apache Software
Foundation\Apache2.2\conf\ssl\ssl.crt\bast_trust_chain.crt".
```

12 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	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
within the last successful HTTP RESPONSE 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 registry1 as long as they also accept "identity" and "gzip" content-coding.	C.9	Basic request / response pattern: 9.	(✓) Deviation: Clients must accept gzip	(✓) Deviation: Clients must accept gzip
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		✗	
Servers SHALL produce and Clients SHALL process the following return codes: - 200 (OK), in responses carrying payload, - 304 (Not Modified), if no payload is send because of the specification in the 'If-Modified-Since' header, - 503 (Service Unavailable), if an active HTTP server is disconnected from the content feed, - 404 (Not Found), if a file based HTTP server does not have a proper payload document stored in the place associated to the URL.	C.15	Additional Rules	(✓) Deviation: 403 instead of 401 Additionally 204 when packet buffer is empty	✓ Additionally 204 when packet buffer is empty
Payload data for Information products SHALL be denoted by a URL according to the following convention: d2lcp_infop = "http://" host [":" port] infop_path "/content.xml" ["?" query] where "infop_path" is a "path" component as specified in section 3.3 of [RFC 2396], but excluding the last path segment.	C.16	Describing payload and interfaces	✗	✗

13 Appendix C - Change notice

Version	Date	Changes
1.2	22.09.2023	• Minor spelling and content errors corrected. • Added new chapters 6.2.1.2 and 7.3.1.2 for the corresponding new interfaces • Supported TLS versions expanded to 1.3
1.2.1	04.03.2024	• Handling of referenced JSON-subschemas adapted in chapter 3 • The Accept-Encoding header value gzip was changed in all request tables from uppercases to lowercases letters • Correction of the Request-URIs in the chapters 6.2.2.1 and 9.2 • Corrected XML-Examples from chapter 8.2.1 • Removed chapter for SNI not supported as SNI is supported by Mobilithek • namespace tag was unified to <soapenv: > in the whole document • SOAP (XML)-Response in chapter 6.1.2.2 adapted
1.2.2	04.04.2024	• Description of a special case for DatexIIv3 SOAP pulls on the data recipient side added in section 7.2.2.1