FINGRID
Datahub

External Interface Specification

15.5.2020

Table of contents

Change log

Date	Version	Changes
13.05.2020	1.1	- Idempotency (2.2.4) - Domains on Peek Message - HTTP compression (2.2.5) - Charset Usage (2.2.6) - Expect: 100-Continue header (2.2.7) - XML Signatures: Sections 1.5, 2.2.2.1 and 2.6 and Appendix A. - Recommendation 8 on message size timeseries
18.01.2019	1.0	First official version

1 Introduction

1.1 Purpose

This document contains the external interface specifications for the B2B channel of the Fingrid Datahub. The CGI Central Market System (CMS) is used to for implementing the B2B channel.

1.2 Scope

The B2B Market Messaging interfaces are the external interfaces used in a B2B context.

B2B Market Messaging services:

- SendMessage operation: interface for Market Participants to send Market Messages/Transactions to the Datahub to have them processed asynchronously;
- ProcessMessage operation: interface for Market Participants to process Market Messages/Transactions in the Datahub (synchronous);
- PeekMessage operation: interface for Market Participants to retrieve a Market Message/Transactions from the Datahub;
- DequeueMessage operation: interface for Market Participants to confirm the acceptance of a received Market Message/transactions from the Datahub

1.3 Target Audience

The target audience for this document are architects, operators, development- and test teams of using the B2B-SOAP interface of the Fingrid Datahub.

1.4 Document Structure

Information

Chapter 1 contains the general information about this document, including its purpose, short description and references to other documents.

Product Standard:

Chapter 2 gives an overview of B2B SOAP channel

Chapter 3 describes the details of the B2B SOAP service operations.

Chapter 4 describes the overview of the SOAP actions and Error Codes.

Fingrid specific:

Chapter 5 describes Fingrid Datahub specific details.

1.5 Document References

Reference	Document (ID)	Version	Date
SOAP	https://www.w3.org/TR/soap12-part1/	1.2	27-04-2007
XSD	https://www.w3.org/standards/techs/xmlschema#stds	1.1	05-04-2012
EVENTS	02-04-09-02 – Datahub Events	1.11.1	02-02-2020
SCHEMAS	Zip file: “ExternalInterfaceSpecification 1.11.1.zip”	1.11.1	
BasicSecurityProfile-v1.1	http://docs.oasis-open.org/ws-brsp/BasicSecurityProfile/v1.1/cs01/BasicSecurityProfile-v1.1-cs01.html	1.1	2014
XML-DSIG-V1.0	https://www.w3.org/TR/2008/REC-xmldsig-core-20080610/	1.1	2008
WSS-SOAP-Message-Security-V1.1.1	http://docs.oasis-open.org/wss-m/wss/v1.1.1/wss-SOAPMessageSecurity-v1.1.1.html	1.1.1	2012
WSS-WSU-V1.0	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	1.0	2004

1.6 Glossary

Term	Description
CMS	CGI Central Market System / Datahub
DEQUEUE	Signoff of a retrieved Message
FIFO	First In First Out
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
PEEK	The retrieval of a Message from the Datahub
PKI	Public Key Infrastructure
UML	Unified Modeling Language
URL	Uniform Resource Locator
UUID	Universally unique identifier
WSDL	Web Service Description Language
XSD	XML Schema Definition Language

1.7 Symbols

The standard UML notation for flow and activity diagrams shall apply.

Text between [] refers to a document in (Document References) Eg [EVENTS]

Text between {} refers to implementation specific parameter in () eg {URL}

1.8 Open Issues

This section contains the identified open issues impacting the External Interface Specification.

#	Description	Decision	Owner
	Technical maximum message size for CMS is 100 MB. Business maximum has to be defined by Fingrid. Big inbound messages will give also big outbound messages. Actors could have issues with processing large XML messages. Better to limit to eg 20 MB.	Default is set to 100 MB. Up to Fingrid to decide, based on testing, to lower this.	Fingrid
	Exact bundling directive for use in sending timeseries	Directive is defined in chapter 5.3. Exact numbers can be changed based on testing.	Fingrid

2 B2B SOAP Channel

This chapter gives an overview of the B2B channel for Market Messaging. The Message Hub provides access to the market functions of the Datahub via web services (B2B) and is the gateway for Market Participants to send and receive messages. These services are executed by the use of standardized messages.

This chapter explains common functionality and mechanisms that are the same for the individual service operations.

2.1 Interface overview

Connectivity through the B2B channel is via a single web service using XML representation of one or more transactions in a SOAP v1.2 envelope (Web Service Definition Language). This web service has four generic operations:

- SendMessage (for aSynchronous processing);
- ProcessMessage (for Synchronous processing);
- PeekMessage;
- DequeueMessage.

In the next paragraphs these are described in context of the different processing scenarios.

2.1.1 Asynchronous Processing

- SendMessage - the message and the transaction(s) inside the message are logged, validated, authenticated and authorised and the Market Participant receives a technical acknowledgement via a synchronous response. When the listed checks have passed the transactions are persisted, after which functional processing of the transaction(s) is done asynchronously from the initial Send action. Functional acknowledgements are available after functional processing through the PeekMessage operation.
- PeekMessage - is used by the Market Participant to retrieve the content of the first message available for it. The Market Participant can then process the message asynchronously and once successful can sign off the message using the DequeueMessage operation.

- DequeueMessage - the Market Participant informs the Datahub that the message has been processed by the Market Participant. If a message is not Dequeued any subsequent PeekMessage will return the same message.

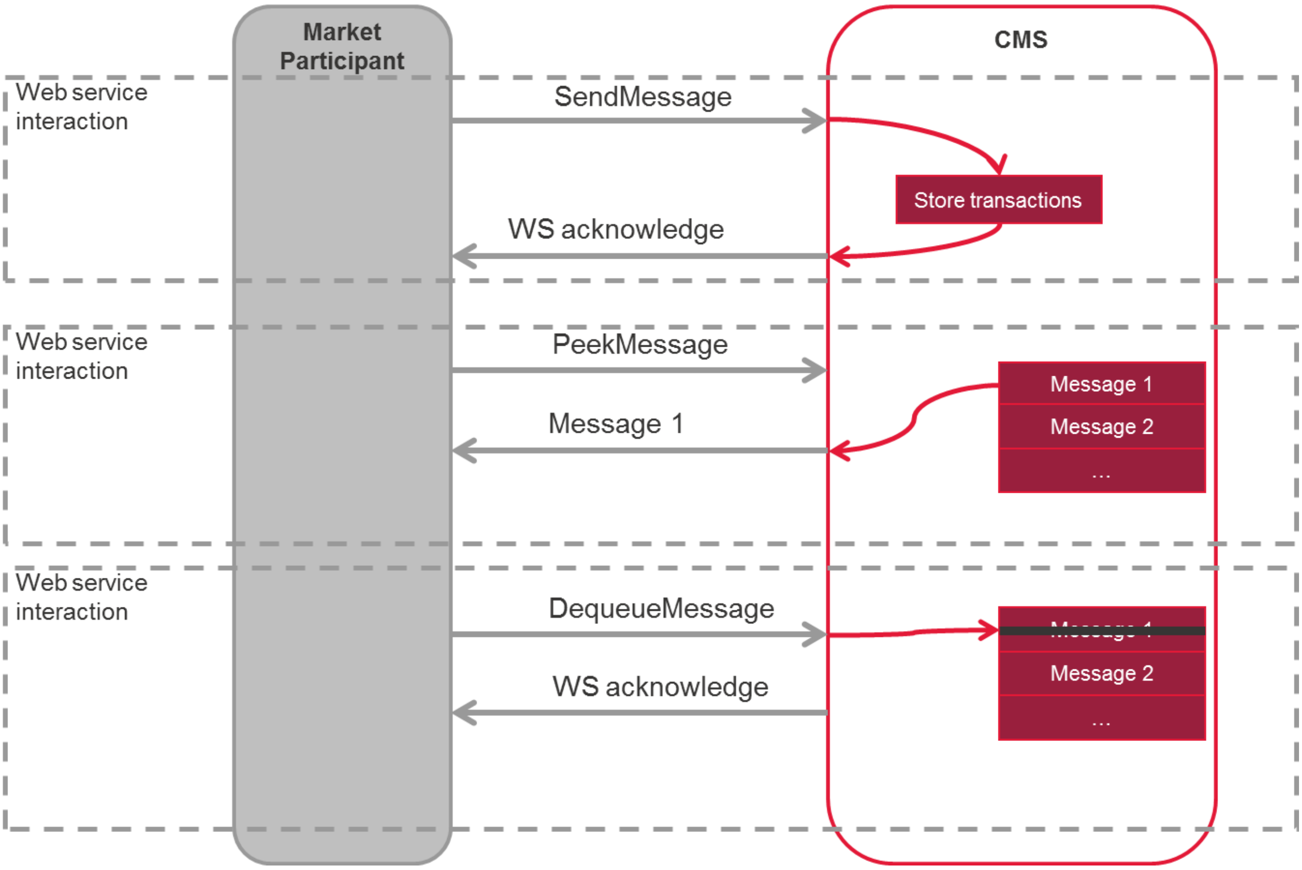
The Send/Peek/Dequeue protocol ensures transactional integrity between the Market Participant and the Datahub.

- In terms of bulk receipt of transactions, the Datahub allows multiple (*) transactions within a single message. The transactions can be of different types (**) and/or for different (**) data elements.
  (*) The number of transactions is limited by:
  - The schema defining the maxOccurs
  - The B2B channel defining the max Message size.

  (**) The schemas define the combinations that that are allowed. Depending on implementation different types are allowed or not.

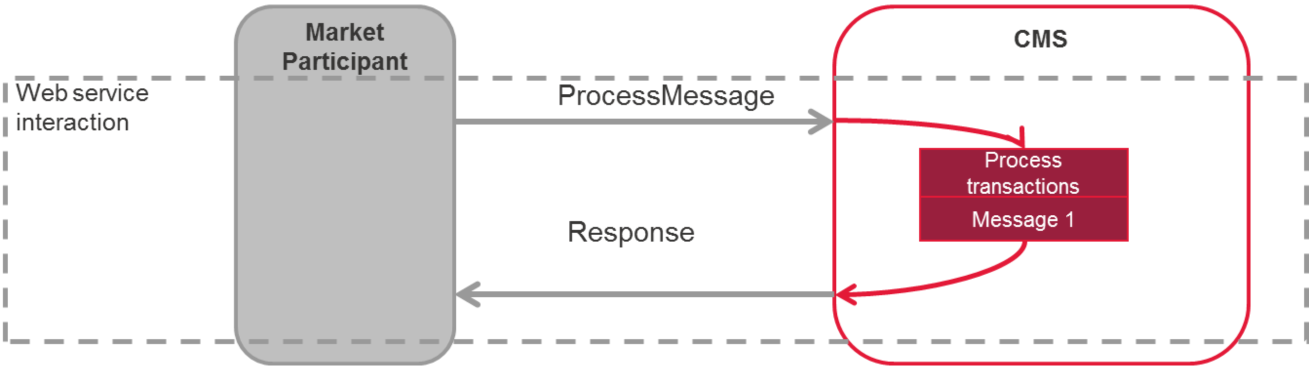
A single synchronous receipt message is sent back to indicate whether the message was accepted or rejected (such as, due to failing XSD validation). Based on the process description the message is either

- Split up and each transaction is processed individually. Any subsequent functional messages (including error messages) are returned to the sending Market Participant individually; as are any follow up messages sent to other Market Participants.
- Processed as a whole. Any subsequent functional messages (including error messages) are returned to the sending Market Participant as group; as are any follow up messages sent to other Market Participants.


Market Participant
CMS
Web service interaction
SendMessage
Store transactions
WS acknowledge
Web service interaction
PeekMessage
Message 1
Message 1
Message 2
...
Web service interaction
DequeueMessage
~~Message 1~~
Message 2
...
WS acknowledge

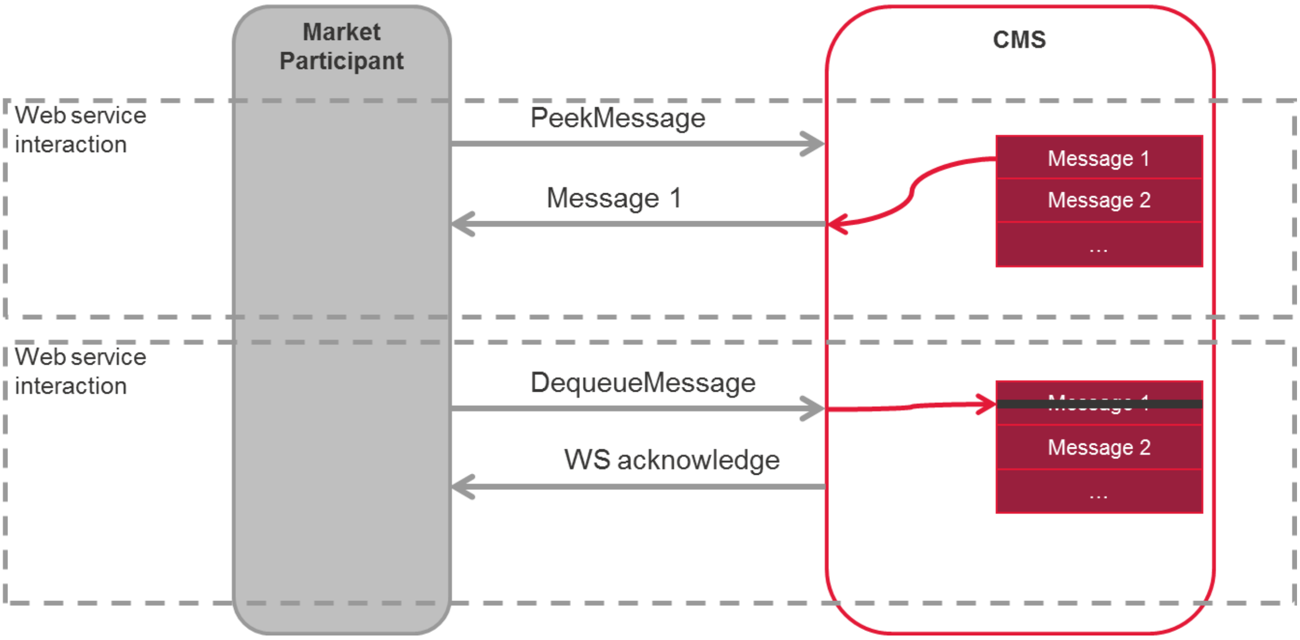
2.1.2 Synchronous Processing

- ProcessMessage - to send transactions to the Datahub. The message and the transaction inside the message are logged, validated, authenticated, authorised, persisted and the transaction is functionally processed, after which the Market Participant receives the result in a synchronous response.



2.1.3 Fetching Notifications

- The Peek and Dequeue protocol is also used in the context of receiving Notification Messages. (Messages for a Market Participant as result of their role in a Market Process, but not directly related to a ProcessMessage from the Market Participant itself).

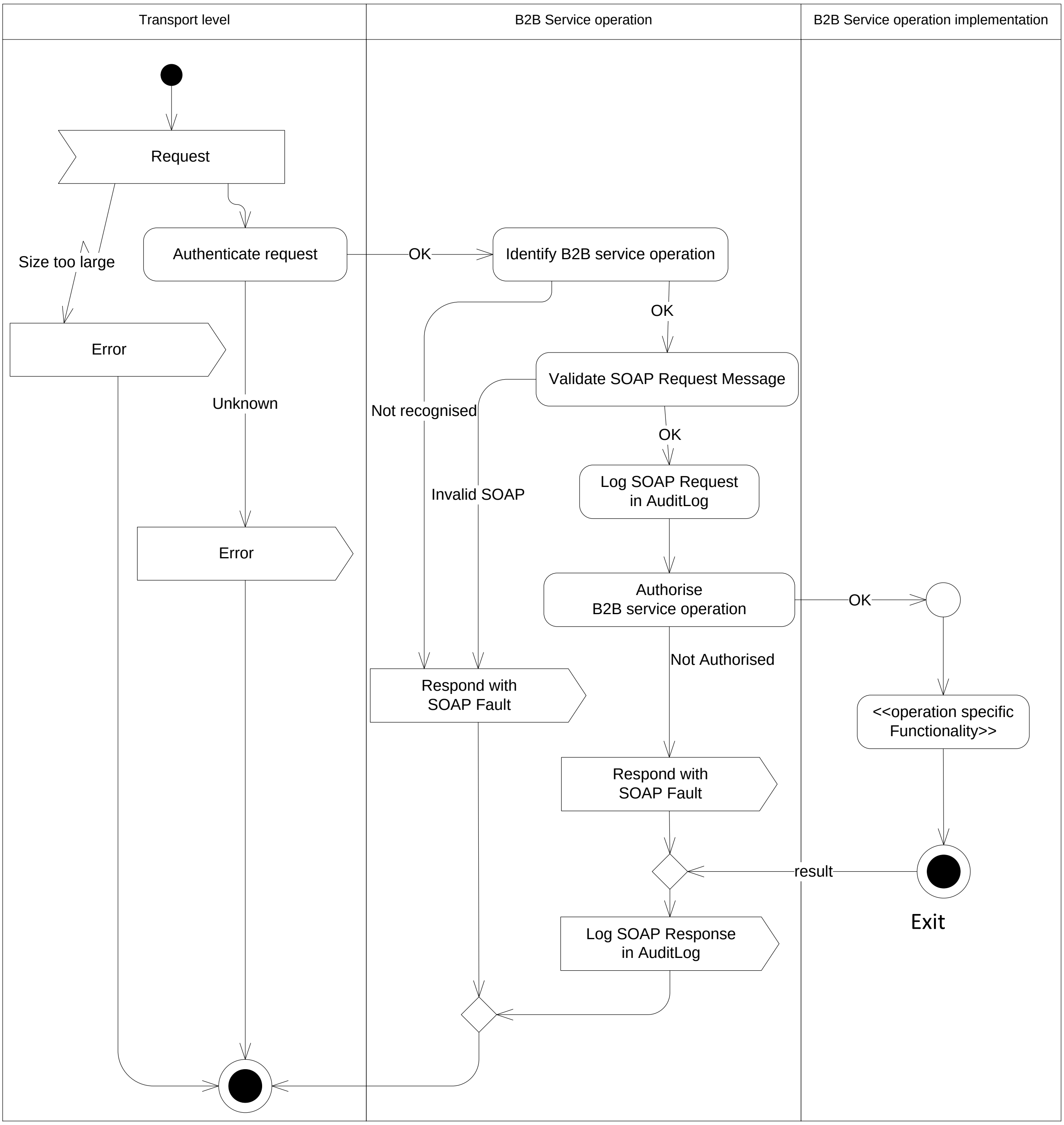


2.2 Generic interface functionality

Each B2B service operation contains functionality that is the same across the operations. This functionality is described here, so this is not replicated for every specific interface.

The generic functionality consists of:

- Transport level
    - Request size checking
    - Transport level authentication
- B2B service operation
    - B2B service operation level authorisation
    - Audit logging of B2B service operation request and response
    - B2B service operation request syntax validation


Transport level
B2B Service operation
B2B Service operation implementation
Request
Size too large
Authenticate request
OK
Identify B2B service operation
Error
OK
Validate SOAP Request Message
Unknown
Not recognised
OK
Log SOAP Request in AuditLog
Invalid SOAP
Error
Authorise B2B service operation
OK
Not Authorised
Respond with SOAP Fault
<<operation specific Functionality>>
Respond with SOAP Fault
result
Exit
Log SOAP Response in AuditLog

The figure above shows the relation between Transport level, Business service operation level and Business service operation implementation, and the steps in each of the areas.

The following sections describe these steps in more detail.

2.2.1 Transport level

The request is received. In case the size of the request is too large an HTTP response with status code 413 (Content too large) is generated and processing is stopped. See Chapter 4.1.1 - Error codes in SOAP Faults

The following error codes are specific to this B2B service operation, besides the error codes mentioned in 2.2.2.1.

Error Code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.007	System	Sender	Unknown or invalid message reference (e.g. cannot dequeue the current message in the MessageQueue if message reference provided does not match the message reference that has been peeked before (i.e. current message))	Sender error. There are two possible casuses: The wrong DocumentReferenceNumber is taken from the Retrieved message. The message has already been Dequeued before (parallel peek and dequeue are not possible).

Summary of error codes for a full list of error responses. The maximum size is configurable, with a maximum of 100 MB.

The authentication of System Users is handled by the Message Hub. The System User is authenticated by a private certificate and can only be used for B2B communication through the web services. If the System User is not authenticated, an HTTP response with status code 401 (Access denied) is generated.

2.2.1.1 Transport level Error codes

The default HTTP codes apply in case of HTTPS transport level faults.

The following HTTP status codes are purposely returned based on the above description.

Error code	Type	Meaning
401	Security	Access Denied – User authentication based on the provided certificate has failed. (check if userIdentity is created correctly)
403	Security	Access Denied – Client certificate is not provided or trusted. (check the certificate in the request)
404	System	Requested resource not found
413	System	Content length too large
500	System	In case of any unidentified errors.

2.2.2 B2B service operation

Based on the request the B2B SOAP operation is identified. In case this is an unknown B2B service operation, a SOAP Fault with code ‘MHB.MED.004’ is returned (See Chapter 4.1.1 - Error codes in SOAP Faults

The following error codes are specific to this B2B service operation, besides the error codes mentioned in 2.2.2.1.

Error Code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.007	System	Sender	Unknown or invalid message reference (e.g. cannot dequeue the current message in the MessageQueue if message reference provided does not match the message reference that has been peeked before (i.e. current message))	Sender error. There are two possible casuses: The wrong DocumentReferenceNumber is taken from the Retrieved message. The message has already been Dequeued before (parallel peek and dequeue are not possible).

Summary of error codes for a full list of error responses).

The syntax of the SOAP Request is checked. When the message syntax is incorrect this results in a SOAP Fault with code ‘MHB.MHD.001’. Syntax checking includes validating the SOAP message using the XML Schema Definitions (XSD’s) and checking the proper grouping of transaction types within the SOAP message.

All the messages that have been authenticated are logged, without change, into the audit log and can be retrieved through the Party User Interface.

The authorisation of the System User for the specific B2B service operation is determined in two steps:

1. If the System User is not found or is not authorized for this B2B service operation then a SOAP Fault is returned with code ‘MHB.MHD.003’ (user not authorized for system function).
2. If the System User neither matches the Organisation specified as sender in the message (PhysicalSender) nor (one of) the delegated organisation(s) a SOAP Fault is returned with code ‘MHB.MHD.009’ (user not authorized for organisation).

Only if all generic checks and validations succeed the request is executed.

In case of an internal technical error the SOAP Fault with code ‘MHB.MHD.000’ is returned.

2.2.2.1 B2B service operation Error codes

The table below provides the error codes that can be returned as a SOAP Fault as part of the generic B2B message handling. Specific B2B message handling is described in chapter 3.

Error code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.000	System	Receiver	General Failure	Resend the message, if issues persists contact the Market Operator
MHB.MHD.001	Syntax	Sender	Message validation failed	Resend corrected message
MHB.MHD.002	System	Receiver	System configuration error	Resend the message, if issues persists contact the Market Operator
MHB.MHD.003	Security	Sender	User not authorized for system function (e.g. not found, no rights for the operation or message type, user blocked or inactive)	Check authorisation and contact Market Operator in case of questions. Resend message after correction of authorisation.
MHB.MHD.004	Security	Sender	Unknown request	Resend corrected message
MHB.MHD.005	System	Receiver	Back-end timeout	Resend the message, if issues persists contact the Market Operator. The system will prevent processing messages with the same transaction-id twice. If the system responses to the resent message with a MHB.MHD.006 (see chapter 3.1.3) then the system has already processed (or is still processing) the first message.

Error code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.008	Security	Sender	Message content unsecure	Message content contains unsecure elements (e.g. SQL injection or cross-site scripting). Message must be adjusted before it can be accepted by the system.
MHB.MHD.009	Security	Sender	User not authorized for organisation (e.g. System User neither matches PhysicalSender nor (one of) the delegated Organisation(s))	Check message header, authorisation and delegation configuration and contact Market Operator in case of questions. Resend message after correction.
MHB.MHD.010	Syntax	Sender	Unknown TenantCode in URL	Correct the TenantCode in the URL and resend the message. See section 2.3
MHB.MHD.011	Syntax	Sender	Unknown System Function	Check message and contact Market Operator in case of questions. Resend message after correction of message content.
MHB.MHD.013	Security	Sender	XML Signature verification failed	Check whether all Signature elements are provided according to specification (see section 2.6) and make corrections where needed before resending the message.

2.2.3 B2B service operation implementation

The specific B2B service operations are not detailed here. See the specific B2B service operations in chapter 3 for more details.

2.2.4 Idempotency check on Message ID’s

As generic principle the Message ID’s (and Transaction ID’s) provided by Market Participants in B2B messages must be unique. In case a provided Message ID or Transaction ID is not unique the CMS rejects the request by responding with a SOAP Fault indicating the Message ID or (one of) Transaction ID(s) was not unique (see Appendix 4.1.1).

It is possible though that the synchronous response sent by CMS to a Market Participant was either not received or not properly processed by the Market Participant. In such a case the Market Participant should be able to request the original synchronous response to allow them to still process it properly.

To support this kind of cases the CMS supports Idempotency for Synchronous processes to allow the request of the original response (deviation to the generic principle):

Idempotency response is designed to ensure completeness of handover of transactions. By sending in the same request (meaning the Message ID will not be unique for CMS) the response to the original message will be returned. This also means that in this case the message won’t be processed any further (i.e. no Business Process will be executed). Idempotency will only work for a limited period of time [(*)], after which CMS will respond with an error message of Duplicate Message ID.

[(*)] This period is controlled by the Market Operator using a Configuration Parameter. The default period is set to 12 hours, but can be set between 0 hour and 1 year, in hourly steps. When set to 0, idempotency is not used.

2.2.5 HTTP Compression

Use the following HTTP header in the request to receive compressed response:

Accept-Encoding: gzip,deflate

Note: If a client uses a proxy server: please ensure it does not remove the “Accept-Encoding” header

2.2.6 Character set

Charset UTF-8 is used in all operations.

2.2.7 Large messages: Expect: 100-Continue HTTP header

As described in section 2.2.1 CMS supports B2B messages up to 100 MB. This maximum size is deployment time configurable, with a maximum of 100 MB.

But for messages larger than ~80 KB there are situations where the http client quickly receives a http error “Connection closed” (or similar depending on the client http implemenation).

To prevent this issue, use the HTTP header “Expect: 100-Continue” while sending messages larger than ~80KB.

Sending the header is applicable to all SOAP requests.

2.3 Exchanging Business Messages

In the following B2B service operations the Business Messages are exchanged:

- SendMessage/ProcessMessage as part of the request;
- PeekMessage as part of the response.

The following URL is used for the posts:

https://<b2b gateway hostname>/soap/<TenantCode>?organisationUser=<organisation username>

The following attributes are provided on the input:

A#	Name	Type	Obligation	Value	Remark
	Connection: Client Certificate	Certificate	Mandatory	The certificate provided by the client.	
	URL: tenantCode	String	Optional	*See chapter 5.1*	
	URL: organisationUser	String	Mandatory	*See chapter 5.1*	The Username is used to uniquely identify the OrganisationUser to be used to do the requests. This allows linking multiple OrganisationUsers to one UserIdentity.
	<Request Body>	String, UTF-8	Mandatory	The message that should be handled by the Datahub.	Request body as described below.

The above B2B service operation requests or responses will always use the MessageContainer structure to wrap/contain the Business Message.

The MessageContainer structure also contains additional information to identify and interpret the Business Message correctly. See Table below.

Element	Obligation	Type	Remark
MessageContainer	Mandatory	Complex Element	
DocumentReferenceNumber	Conditional	xs:string max=36	The DocumentReferenceNumber (i.e. UUID) is used to identify the data transfer of a Business Message for Peek and Dequeue of Messages (only available if part of a response message).
Payload	Mandatory	Complex Element	Contains the actual Business Message in XML message format. Payload should comply with the market message definition and should include the correct namespace.

2.3.1 XML Business Messages in the Payload

XML Business Messages are XML based and always have a single root element. This means that the entire Business Message is placed in the Payload element. If there are multiple transactions in the Business Message they are part of the same payload.

The namespace of the specific XML Business Message must be included in the message. This can be on the Business Message level (also known as locally defined), but also on a higher level (also known as globally defined); as long as the namespace is defined in the message.

This is based on and in line with [SOAP] and [XSD].

2.4 System Users

For the System User a User Identity is set up that is identified by a client certificate (PKI) for its authentication. This User Identity can be linked to one or more Organisation Users within the Datahub. The System Users can only be used for B2B communication through the B2B services.

According to the figure 1 below, the System User (User Identity) is related to an Organisation using the Organisation User entity and System User acts on behalf of that Organisation. The same User Identity can be linked to Organisation Users for different Organisations. An Organisation can as such have multiple System Users.

A System User can only execute transactions for the specific Organisations the System User is related to.

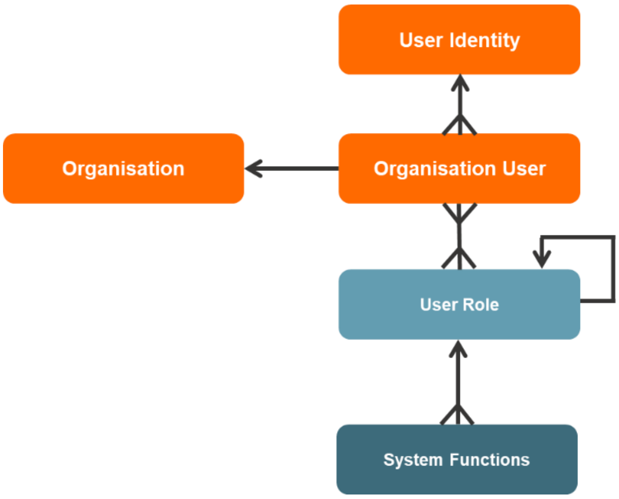


FIGURE 1: RELATION BETWEEN USER INDENTITY, ORGANIASTION USER, ORGANISATION AND USER ROLES AND SYSTEM FUNCTIONS

Authorization of a System User is performed at 2 levels based on the User Roles attached to the linked Organisation User:

- B2B Service operation: is the System User authorized to access this B2B service operation?
- Business Messages: is the System User authorized to send in this type of Business Message on behalf of the Organisation (or the delegating Organisation)?

2.5 SOAP Fault details

The SOAP fault message will have a limited set of information. The default SOAP fault is returned. The table below shows the elements in a SOAP fault message, where the table indents indicate child elements of a Complex Element. The table below also indicates the origin of the elements, either generic SOAP standard (SOAP) or specific to the CMS product.

Element	Obligation	Origin	Notes
Fault	Mandatory	SOAP	Complex Element
Code	Mandatory	SOAP	Complex Element. Indicates whether the fault is caused by the Sender or Receiver.
Value	Mandatory	SOAP	"**Sender**" or "**Receiver**" These values contain a prefix to the http://www.w3.org/2003/05/soap-envelope namespace. **Sender**: The message was incorrectly formed or did not contain the appropriate information in order to succeed. It is generally an indication that the message is not to be resent without change **Receiver**: The message could not be processed for reasons attributable to the processing of the message rather than to the contents of the message itself. For example, processing could include communicating with an upstream SOAP node, which did not respond. The message could succeed if resent at a later point in time.
Subcode	Optional	SOAP	Not used
Reason	Mandatory	SOAP	Complex Element. Provides a human-readable explanation of the fault.
Text	Mandatory	SOAP	Contains the text of the human-readable explanation of the fault. Must contain an attribute specifying the language with local name "**lang**" in the namespace http://www.w3.org/XML/1998/namespace
Node	Optional	SOAP	Not used
Role	Optional	SOAP	Not used
Detail	Mandatory	SOAP	Complex Element. Elements below are specific to CMS product.
CMSFault	Mandatory	CMS	Complex Element. Root element for the specific CMS product Fault elements.
ErrorCode	Mandatory	CMS	Error Code (e.g. MHB.MHD.001) identifying specific error situation (See chapter 4.1.1).
ErrorIdentification	Optional	CMS	The number as logged in CMS. Based on the Error identifier the full details regarding the error can be obtained at the Market Operator.
ErrorDetails	Conditional	CMS	Details about the cause of the error (only if XML validation failed due to message syntax errors).

2.5.1 Classifying Error codes in this document

In this document the error codes are classified to provide more insight. The classifications are:

- Security error; related to authentication, authorization issues
- Syntax error; related to syntactically incorrect messages or Business Messages
- System error; related to technical errors in CMS

2.5.2 SOAP Fault example

Below is an example SOAP Fault:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:urn="urn:cms:b2b:v01">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <SOAP-ENV:Code>
            <SOAP-ENV:Value>SOAP-ENV:Sender</SOAP-ENV:Value>
         </SOAP-ENV:Code>
         <SOAP-ENV:Reason>
            <SOAP-ENV:Text xml:lang="en">Message validation failed</SOAP-ENV:Text>
         </SOAP-ENV:Reason>
         <SOAP-ENV:Detail>
            <urn:CMSFault>
               <urn:ErrorCode>MHB.MHD.001</urn:ErrorCode>
               <urn:ErrorIdentification>1540987503286</urn:ErrorIdentification>
               <urn:ErrorDetails>javax.xml.stream.XMLStreamException: org.xml.sax.SAXParseException;
lineNumber: 7; columnNumber: 49; cvc-datatype-valid.1.2.1: '?' is not a valid value for
'dateTime'.</urn:ErrorDetails>
            </urn:CMSFault>
         </SOAP-ENV:Detail>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.6 XML Message signing

Datahub market group has decided that XML message signing will not be taken into use at Datahub go-live in 2022. Datahub will support message signing, as described below, but vendors do not need to implement the XML signing to their systems until instructed otherwise.

The purpose of XML signatures is to verify that a given XML document is authentic and integral. An authentic document is one which was actually signed by the party claiming to have signed it. An integral document is one which hasn't been modified in the process of sending the document to the receiver. Use of XML signatures also provides non-repudiation as the sender cannot challenge the validity of the message or its origin (at least as long as the sender's private key is not compromised). Note that XML signatures do not address message encryption (CMS relies on SSL/TLS for this).

CMS adheres to the following standards and specifications with regards to WS-Security and XML Signatures (references can be found in section 1.5):

- BasicSecurityProfile-v1.1
- XML-DSIG-V1.0 (prefix DS)
- WSS-SOAP-Message-Security-V1.1.1 (prefix WSSE)
- WSS-WSU-V1.0 (prefix WSU)

Any and all limitations and recommendations that CMS implemented on top of the above standards and specifications are provided below:

1. The XML Signatures must be placed in the Header element of the SOAP Envelope, where the SOAP Body element contains the Business Message. That way the Signature can always be handled separately from the signed content, (so called Detached Signature).
2. It is recommended to sign the entire SOAP Body to mitigate XML Signature wrapping. The Signature contains one or more references to the SOAP Body element (or subelements of SOAP Body) to indicate the scope of what is signed. It is important to choose a proper scope, where all data is included that must be signed and any data that may be changed is excluded.

   For the Finnish Datahub the SendMessage (section 3.2) and ProcessMessage (section 3.1) supports the use of the WSU Id attribute on the 'root' node (element below the Payload element) of the Business Message.
3. It is recommended to use Exclusive Canonicalization as this only includes the namespaces in the Signature that are actually used, thereby preventing the Signature to become invalid if the signed scope is moved into another XML document (e.g. when the CMS namespace 'urn:cms:b2b:v01' is not used when signing the Business Message but added later in order to create a request message according to the CMS WSDL as this will wrap the Business Message into a ProcessMessage or SendMessage element, see chapter 3).
4. The use of WSU Ids to reference the signed scope is required because they are WS-I compliant. With WSU Ids, an Id attribute is inserted into the root element of the scope/nodeset that is to be signed. The XML Signature then references this Id to indicate the nodes that were signed.
5. Binary tokens (asymmetric encryption) must be used instead of username token as this enhances security by not sharing private keys between parties, where the requestor's certificate (X509v3) must be used for creating the signature and the public key for that certificate must be included in the request. A recommendation on top of this is to send the public key (certificate) using BinarySecurityToken (and Reference) or KeyIdentifier instead of only the X509Data in order to also specify the certificate type (e.g. v1 or v3) and the encoding that was applied (e.g. Base64).
6. It is recommended not to use multiple Signatures or sign multiple sections of the message as this increases the message size and increases the overhead of Signature computation. If multiple separate parts in the SOAP Body need to be signed, the recommended approach is to use multiple Reference elements within the same Signature opposed to creating separate Signatures for each SOAP Body part to sign. This reduces the overhead in both message size and Signature computation.
7. It is recommended not to include the WSU Timestamp element. It can be used to protect against replay attacks, but CMS already has a more strict protection against this by using unique message identifiers (see section 2.2.4). Including the WSU Timestamp element requires a separate Reference in the Signature with its own digest which increases the overhead (larger message and additional Signature processing) as this ensures the integrity of

the Timestamp values. If the Signature has expired (Expires element date/time is before the CMS system date/time) the Signature (and message) is rejected (see below SOAP Fault details).

8. To verify the authenticity of the XML Signatures the JuridicalSender of the message must also have the certificate used for signing the message configured within CMS, so that the provided public key can be compared to the configured public key. If these don't match it is possible the Signature was 'forged' by a third party so CMS will reject the XML Signature (and the message). Another possibility is that the certificate is not correctly configured in CMS (e.g. because it has expired and the Organisation is using a new certificate to sign the message), so ensure the certificate is kept up to date in CMS configuration.

   In the case of Message Delegation where the Delegated Organisation (= Physical Sender) signs the message it is the responsibility of the Delegating Organisation (=Juridical Sender) to also include the public key of the Delegated Organisation in their own public keys configuration in CMS.
9. WS-Security recommends using Shorthand XPointers to the data, meaning they are referenced on WSU Id, as opposed to referencing the data only using XPath. XPath can only be used as a filter on top of the wsu id. Ensure to use XPath Filter 2.0 (http://www.w3.org/2002/06/xmldsig-filter2) as this is more efficient than the original XPath transform (http://www.w3.org/TR/1999/REC-xpath-19991116).
10. Although the standards still describe SHA-1 as an acceptable Signing algorithm it must no longer be used to sign messages due to collisions, where the signature can be duplicated and forged. Use SHA-256 instead.

If CMS Signature verification fails a SOAP Fault is returned with ErrorCode MHB.MHD.013 and Text "XML Signature verification failed" (see section 2.2.2.1). See Appendix C for a list of XML Signature examples.

The table below provides an overview of the elements that can be provided when signing a SOAP message:

Element	Cardinality	Origin Prefix	Notes
Security	0..1	WSSE	Must be in the namespace 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'.
Timestamp	0..1	WSU	See recommendation 7 above. Defines the creation time of the message and how long the message is considered valid. An Id property can be defined that gives the 'Timestamp' element a unique name so that it can be used later in the 'Signature' element to reference this 'Timestamp' element (to include the timestamp in the digital signature). Must be in the namespace 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'.
Created	1..1	WSU	Date and time to indicate when the Signature was created.
Expires	0..1	WSU	Date and time to indicate when the Signature expires. If the time of processing in CMS is after this date and time the Signature is considered invalid.
BinarySecurityToken	0..1	WSSE	Contains the base-64 encoded X509v3 public signer certificate that corresponds to the private key that was used to generate the digital signature. **Must be provided** if Reference is used in KeyInfo (see below). **Must have** an attribute names 'ValueType' with value specified by the related security token profile (e.g. http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3). **Must have** an attribute named 'EncodingType' with value 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary'.
Signature	1..n	DS	Must be in the namespace 'http://www.w3.org/2000/09/xmldsig#'. May have attribute 'Id', which is mandatory in case multiple Signature elements are used.
SignedInfo	1..1	DS	Wrapper element for the signature's meta-data and signed data.
CanonicalizationMethod	1..1	DS	Attribute 'Algorithm' specifies the canonicalization algorithm applied to the SignedInfo element prior to performing signature calculations. To avoid namespace issues the exclusive canonicalization algorithm should be used (http://www.w3.org/2001/10/xml-exc-c14n#).
SignatureMethod	1..1	DS	Attribute 'Algorithm' defines the algorithm that is used to produce the digital signature (http://www.w3.org/2001/04/xmldsig-more#rsa-sha256). Asymmetric encryption (RSA) is used as well as hashing (SHA).
Reference	1..n	DS	Specifies a digest algorithm and digest value, and an required identifier of the object being signed, the type of the object, and/or a list of transforms to be applied prior to digesting.
Transforms	1..1	DS	Contains a series of transforms that should be applied to the data being signed, where the attribute 'Algorithm' specifies the transform logic to apply. At least one transform element must exist (e.g. with Algorithm= 'http://www.w3.org/2001/10/xml-exc-c14n#').
Transform	1..n	DS	Attribute 'Algorithm' specifies the transform logic to apply before signing or verifying the signature.
XPath	0..n	DS	Optionally XPath references can be provided to indicate the scope/nodes to be signed inside the scope identified by the refrence identifier (wsu:id). This is particularly useful when certain nodes

			within the scope should be excluded from the signature (e.g. because these nodes are allowed to be altered after signing).
DigestMethod	1..1	DS	Defines the hashing algorithm that is used on the data described by the URI property in the attribute ‘Algorithm’. Use one of: • http://www.w3.org/2000/09/xmldsig#sha1 • http://www.w3.org/2001/04/xmlenc#sha256 • http://www.w3.org/2001/04/xmldsig-more#sha384 • http://www.w3.org/2001/04/xmlenc#sha512
DigestValue	1..1	DS	The calculated hash value of the input data after: 1. Canonicalization (see Transform attribute ‘Algorithm’) 2. Applying the hashing algorithm specified by DigestMethod 3. Base64 encoding
SignatureValue	1..1	DS	Base64 encoded digital signature of the ‘SignedInfo’ element (after the canonicalization algorithm described by ‘CanonicalizationMethod’ element is applied).
KeyInfo	0..1	DS	Contains information about the key that was used to produce the digital signature.
SecurityTokenReference	1..1	WSSE	Mandatory element for WS-Security.
KeyIdentifier	0..1	WSSE	Contains the public part of the certificate used to sign the Signature. Attribute EncodingType indicates the encoding of the certificate (e.g. Base64) and attribute ValueType indicates the type of certificate (e.g. ‘http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3’).
Reference	0..1	WSSE	Reference to BinarySecurityToken, set by the ‘URI’ attribute. The ‘ValueType’ attribute must be set to the security token profile (e.g. ‘http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3’). This is particularly useful when multiple Signatures are created using the same certificate as the certificate itself is only defined once then (in BinarySecurityToken) opposed to defining it again in each Signature (in KeyIdentifier).
X509Data	0..1	DS	Contains the public part of the certificate used to sign the Signature. Not recommended as this element does not specify the certificate type (e.g. v1 or v3) and also doesn’t specify the encoding that was applied (e.g. Base64).

2.7 Timestamps

Datahub logs the timestamp for each incoming operation request and each outgoing operation response. These timestamps are logged in the system and used for reporting. Any timestamps in the message and transactions as provided by the Market Participant are stored for reference but not used for operational management or reporting. All timestamps in B2B communication are presented in UTC and in XML format YYYY-MM-DDTHH:MM:SS+00:00 or YYYY-MM-DDTHH:MM:SSZ. All timestamps that are stored in the system are stored in UTC.

3 B2B SOAP Service operation

This chapter describes the external interfaces used in a B2B context.

3.1 ProcessMessage operation

A System User uses ‘ProcessMessage’ to send XML Business Messages concerning a market process to the Datahub that needs synchronous processing. The operation implementation directly executes the Market processes and returns the processing result (i.e. synchronously).

3.1.1 Request

Element	Obligation	Type	Notes
ProcessMessageRequest	Mandatory		
MessageContainer	Mandatory	Complex Element	
Payload	Mandatory	Complex Element	Contains the actual Business Message in XML message format. Payload should comply with the market message definition and should include the right namespace (see section 2.3).

The following rules are enforced:

- The Payload is syntactically correct (validated against the XSD)
- The System User is authorized to send in the message (Document Type) for the Sender mentioned in the Business Message
- The Message and Transaction ID’s used in the message are unique (i.e. not used before).

Failing these rules will result in a corresponding SOAP Fault.

Process validations errors will result in a Response message and not in a SOAP fault.

3.1.2 Response

Element	Obligation	Type	Notes
ProcessMessageResponse	Mandatory		
MessageContainer	Mandatory		
DocumentReferenceNumber	Mandatory	xs:string max=36	The DocumentReferenceNumber (i.e. UUID) as generated by the Datahub is used to identify the data transfer of a Business Message for Peek and Dequeue of messages.
Payload	Mandatory	Complex Element	Contains the actual Business Message in XML message format. Payload should comply with the market message definition and should include the right namespace (see section 2.3).

3.1.3 Error codes in SOAP Faults

The error codes in the table below are specific to the ProcessMessage B2B service operation, besides the generic error codes mentioned in paragraph 2.2.2.1.

Error Code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.006	Syntax	Sender	The provided Ids are not unique or have been used before	Two possible issues, for both resending is not applicable: Issue in the client system which results in sending messages with the same id's Retry after a MHB.MHD.005 (see operational guidance on that error)

3.2 SendMessage operation

A System User uses 'SendMessage' to send XML Business Messages concerning a market process to the Datahub. The operation implementation does not directly execute the Market processes but only receives, checks the syntax, authorisation, uniqueness of ID's and forwards the Business Message internally. Therefore the response is only a technical acknowledgement to indicate that the request will be processed. The actual processing of the Business Messages is done internally (i.e. asynchronously).

3.2.1 Request

Element	Obligation	Type	Notes
SendMessageRequest	Mandatory		
MessageContainer	Mandatory	Complex Element	
Payload	Mandatory	Complex Element	Contains the actual Business Message in XML message format. Payload should comply with the market message definition and should include the right namespace (see section 2.3).

The following rules are enforced:

- The Payload is syntactically correct (validated against the XSD)
- The System User is authorized to send in the message (Document Type) for the Sender mentioned in the Business Message
- The Message and Transaction ID's used in the message are unique (i.e. not used before).

Failing these rules will result in a corresponding SOAP Fault.

3.2.2 Response

Element	Obligation	Type	Notes
SendMessageResponse	Mandatory		
DocumentReferenceNumber	Mandatory	xs:string max=36	UUID

3.2.3 Error codes in SOAP Faults

The error codes in the table below are specific to the SendMessage B2B service operation, besides the generic error codes mentioned in paragraph 2.2.2.1.

Error Code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.006	Syntax	Sender	The provided Ids are not unique or have been used before	Two possible issues, for both resending is not applicable: Issue in the client system which results in sending messages with the same id's Retry after a MHB.MHD.005 (see operational guidance on that error)

3.3 PeekMessage operation

A System User uses 'PeekMessage' to retrieve a Business Message from the Datahub. It returns the top most message (FIFO) on the logical queue for the Market Participant that has not been marked as Dequeued. These business messages can contain several transactions.

Please note that PeekMessage returns a Business Message that shall be processed by the caller of PeekMessage, without dequeuing (see section **Error! Reference source not found.**) this business message first.

It is the responsibility of the Market Participant to regularly peek, process and dequeue messages. The Datahub will continue processing and prepare additional messages independent of the Market Participant retrieving messages. Messages are delivered in the order that the Datahub has created the messages. Datahub has reporting to monitor the size of the Message Queues. Datahub does not retire messages in the Message Queue.

When no message is found on the queue the response does not contain a MessageContainer element.

3.3.1 Request

Element	Obligation	Type	Notes
PeekMessageRequest	1..1	Complex Element	
MessageDomains	0..1	Complex Element	

MessageDomain	1..6	xs:string
		max=4
		
		Domain:
		BB01=Structural data
		BB02=Metering data (DH-211 and DH-212)
		BB03=Invoicing information
		BB04=Settlement Data (DH-500)
		BB05=Balance Deviation Data (DH-600)
		BB06=Metering Data (DH-22x)
			Use MessageDomain to filter the MessageQueue, if provided in the request (Market Participant is able to peek messages per domain).

3.3.2 Response

Element	Obligation	Type	Notes
PeekMessageResponse	1..1	Complex Element	
MessageContainer	0..1	Complex Element	Only if Business Message found on the queue.
DocumentReferenceNumber	1..1	xs:string max=36	The DocumentReferenceNumber (i.e. UUID) as generated by the Datahub to identify the data transfer of a Business Message, which should be used for subsequent Dequeue of this message.
Payload	1..1	Complex Element	Contains the actual Business Message in XML message format. Payload should comply with the market message definition and should include the right namespace (see section 2.3).

A Peek Message is acknowledged by a Dequeue Message containing the DocumentReferenceNumber sent in the header of the Peek Message. As a consequence, a new PeekMessage before dequeue will result in resending the last retrieved message. This allows for a Market Participant to receive the same message for processing in case processing failed on their end. All retrieved messages are registered in the audit log.

3.3.3 Error codes in SOAP Faults

There are no additional error codes for this B2B service operation, besides the error codes mentioned in 2.2.2.1.

3.3.4 Operational guidance

In case the PeekMessage response does not contain a payload, the Market Participant system should implement a wait cycle before issuing a new PeekMessage request. See section 0 for wait time. This also applies where the system returns an internal server error.

3.4 Dequeue Message operation

A System User uses 'DequeueMessage' to indicate to the the Datahub it has handled a message it retrieved from the Datahub using 'PeekMessage' and has been able to extract the content in order to use the DocumentReferenceNumber of the business message to sign off. This message will be removed from the logical queue for this Market Participant. Note that when this is done the next message becomes available on the logical queue for retrieving.

It is the responsibility of the Market Participant to regularly peek, process and dequeue messages. The Datahub will continue processing and prepare additional messages independent of the Market Participant retrieving messages. Messages are delivered in the order that the Datahub has created the messages.

3.4.1 Request

Element	Obligation	Type	Notes
DequeueMessageRequest	Mandatory		
DocumentReferenceNumber	Mandatory	xs:string max=36	UUID

Constraints

Element	Constraint	Error code
DocumentReferenceNumber	Must be the DocumentReferenceNumber in the Business Message from the previously peeked message (see PeekMessage).	MHB.MHD.007 (see below)

3.4.2 Response

Element	Obligation	Type	Notes
DequeueMessageResponse	Mandatory		Empty element.

4 Summary

4.1 Summary of SOAP actions

The following SOAP Actions have to be used in the requests

Service operation	SOAP Action
SendMessage	sendMessage
ProcessMessage	processMessage
PeekMessage	peekMessage
DequeueMessage	dequeueMessage

4.1.1 Error codes in SOAP Faults

The following error codes are specific to this B2B service operation, besides the error codes mentioned in 2.2.2.1.

Error Code	Type	SOAP Code	Meaning	Operational Guidance
MHB.MHD.007	System	Sender	Unknown or invalid message reference (e.g. cannot dequeue the current message in the MessageQueue if message reference provided does not match the message reference that has been peeked before (i.e. current message))	Sender error. There are two possible casuses: The wrong DocumentReferenceNumber is taken from the Retrieved message. The message has already been Dequeued before (parallel peek and dequeue are not possible).

4.2 Summary of error codes

This section provides a summary of all error codes that may occur.

Transport level error codes		
Error code	Type	Meaning
401	Security	Access Denied – .User authentication based on the provided certificate has failed.(check if userIdentity is created correctly)
403	Security	Access Denied – Client certificate is not provided or trusted.(check the certificate in the request)
404	System	Requested resource not found
413	System	Content length too large
500	System	In case of any unidentified errors.
Message handling error codes		
Error code	Type	Meaning
MHB.MHD.000	System	General Failure
MHB.MHD.001	Syntax	Message validation failed
MHB.MHD.002	System	System configuration error
MHB.MHD.003	Security	User not authorized for system function (e.g. not found, no rights for the operation or message type, user blocked or inactive)
MHB.MHD.004	Security	Unknown request
MHB.MHD.005	System	Back-end timeout
MHB.MHD.006	Syntax	The provided Ids are not unique or have been used before (MessageID and TransactionID)
MHB.MHD.007	System	Unknown or invalid message reference (e.g. cannot dequeue the current message in the MessageQueue if message reference provided does not match the message reference that has been peeked before (i.e. current message))
MHB.MHD.008	Security	Message content unsecure
MHB.MHD.009	Security	User not authorized for organisation (e.g. System User neither matches PhysicalSender nor (one of) the delegated Organisation(s))
MHB.MHD.010	Syntax	URL not correct
MHB.MHD.011	Syntax	Unknown System Function
MHB.MHD.013	Security	XML Signature verification failed

5 Fingrid Datahub – WSDL details

This section contains Fingrid specific recommendations, file descriptions and examples.

5.1 Parameters

Parameters for Fingrid

Parameter	Value
{TenantCode}	FGR
{URL}	Will be communicated separately.
{OrganisationUser}	Specfic per Organisation. Defines the registered b2b user in the Datahub created by Fingrid having specific UserRoles for B2B communication.
Max message size	TBD (see open issue #3)

5.2 Connection details

Parameter	Value
Protocol	HTTPS (TLS 1.2)
Port	443
SOAP Version	1.2
Style / use	Document / Literal

5.3 Recommendations

Recommendations for the use of the B2B-SOAP interface.

A system not operating according to these recommendations is considered to be using the interface incorrectly.

#	Recommendations	Reasoning
	UTF-8 shall be used for the XML messages Content-Type: text/xml; charset=UTF-8	Preventing encoding issues
	Prevent pretty print.	Pretty printing gives overhead of quite some whitespace.
	Use of UUID in Message and Transaction ID's	Ensuring uniqueness of identifiers
	Prevent sending messages for same accounting point / agreement/ customer at the same time.	Datahub does support processing messages in parallel (Send/Process) but can't not guarantee a sequence in processing when messages are send in parallel
	Don't use Peek/Dequeue in parallel	Datahub does not support parallel Peek messages for a domain. Parallel request will result in getting same message multiple time
	Polling: When Peek message does not return a message sleep for 30 sec seconds before retry.	Prevent too much requests in time window.
	Use HTTP Compression when requesting (PeekMessage) data from Datahub	Better response times, less network traffic.
	Note on message size of time series data In order to have effective processing in the Datahub, a recommended message size range is applied due to the following constraints: Too many too small messages will give a huge overhead on sending (DSO) and retrieving (supplier, third parties) messages; Too large messages will give too much delay in processing, increased system resources limiting the ability to process messages from different senders in parallel	Minimum size: For the minimum size, DSOs shall – for normal reporting purposes – collect the data for at least 100 metering points of daily (24 hourly values) time series. In theory this would result in 35.000 messages per day in total. In case of reporting corrections to time series data, this minimum size is not applicable if meeting the deadline for sending in time series would otherwise not be met. Recommended size: The recommended size for reporting is defined to be between 500 and 2000 metering points of daily (24 hourly values) time series. There will be no hard enforcing of these rules by rejecting small messages, but this will be monitored.
	Apply recommondations stated in chapter 2.6 on message signing	See chapter 2.6
	During most test phases Datahub will not reject messages without a signature header. Start using the signature header during test to prevent issues in later stages as during production(test) the siging header will be required.	Facilitate testing.

5.4 File descriptions

The following table describes the folder structure of the schemas delivery:

Folder	Content
\common	general data elements
\invoicing	message structure for the domain invoicing
\masterdata	message structure for the domain masterdata
\metering	message structure for the domain metering
\query	message structure for the domain query
\wsdl	WSDL for use in communication with Datahub
\test	WSDL for test preparation purposes only. Supports easier setup of test messages in tools like SOAP-UI.

5.5 Fingrid Datahub Examples

The examples are for showing correct structure of the messages. The used values for elements do apply the schema validation rules but are not based on real data.

The examples are based on the [EVENTS] document. This chapter does not include the signing information.

See for details on signing: Appendix A.

5.5.1 Send Message Request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01"
xmlns:urn2="urn:fi:Datahub:mif:common:HDR_Header:elements:v1" xmlns:urn3="urn:fi:Datahub:mif:common:PEC_ProcessEnergyContext:elements:v1"
xmlns:urn7="urn:fi:Datahub:mif:invoicing:F13_InvoicingDataInfo:v1" xmlns:urn8="urn:fi:Datahub:mif:invoicing:F13_InvoicingDataInfo:elements:v1">
  <soap:Header/>
  <soap:Body>
    <urn:SendMessageRequest>
      <urn:MessageContainer>
        <urn:Payload>
          <urn7:InvoicingDataInfoMessage>
            <urn7:InvoicingDataInfo>
              <urn7:Header>
                <urn2:Identification>345345345345345</urn2:Identification>
                <urn2:DocumentType>F13</urn2:DocumentType>
                <urn2:Creation>2018-01-01T10:10:00</urn2:Creation>
                <urn2:PhysicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">8700000000001</urn2:Identification>
                </urn2:PhysicalSenderEnergyParty>
                <urn2:JuridicalSenderEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">8700000000001</urn2:Identification>
                </urn2:JuridicalSenderEnergyParty>
                <urn2:JuridicalRecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">8700000000002</urn2:Identification>
                </urn2:JuridicalRecipientEnergyParty>
                <urn2:PhysicalRecipientEnergyParty>
                  <urn2:Identification schemeAgencyIdentifier="9">8700000000002</urn2:Identification>
                </urn2:PhysicalRecipientEnergyParty>
                <urn2:SenderRoutingInformation>rtyrtyrty</urn2:SenderRoutingInformation>
              </urn7:Header>
              <urn7:ProcessEnergyContext>
                <urn3:EnergyBusinessProcess>DH-731-1</urn3:EnergyBusinessProcess>
                <urn3:EnergyBusinessProcessRole>DDQ</urn3:EnergyBusinessProcessRole>
                <urn3:EnergyIndustryClassification>23</urn3:EnergyIndustryClassification>
              </urn7:ProcessEnergyContext>
              <urn7:Transaction>
                <urn8:MeteringPoint schemeAgencyIdentifier="9">123456789012345678</urn8:MeteringPoint>
                <urn8:Contract>43456</urn8:Contract>
                <urn8:PartyIdentification schemeAgencyIdentifier="9">456546456</urn8:PartyIdentification>
                <urn8:InvoicingPeriod>
                  <urn8:Start>2018-01-01T10:10:00</urn8:Start>
                  <urn8:End>2018-01-01T10:10:00</urn8:End>
                </urn8:InvoicingPeriod>
                <urn8:InvoicingRow>
                  <urn8:ProductIdentification>345345</urn8:ProductIdentification>
                  <urn8:ProductComponentIdentification>23452345</urn8:ProductComponentIdentification>
                  <urn8:Price>000000099999999.99999900000</urn8:Price>
                  <urn8:PriceUnit>EUR</urn8:PriceUnit>
                  <urn8:PriceUnitCode>EUR</urn8:PriceUnitCode>
                  <urn8:Currency>EUR</urn8:Currency>
                  <urn8:TaxIncluded>1</urn8:TaxIncluded>
                  <urn8:Volume>1.887</urn8:Volume>
                  <urn8:VolumeUnit>KWH</urn8:VolumeUnit>
                  <urn8:VolumeUnitCode>EUR</urn8:VolumeUnitCode>
                  <urn8:Amount>12.99</urn8:Amount>
                  <urn8:Description>Thisisatest</urn8:Description>
                  <urn8:Tax>0</urn8:Tax>
                  <urn8:Start>-2018-01-01T10:10:00</urn8:Start>
                  <urn8:End>2018-01-02T10:10:00</urn8:End>
                </urn8:InvoicingRow>
              </urn7:Transaction>
            </urn7:InvoicingDataInfo>
          </urn7:InvoicingDataInfoMessage>
        </urn:Payload>
      </urn:MessageContainer>
    </urn:SendMessageRequest>
  </soap:Body>
</soap:Envelope>
```

5.5.2 Send message response

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
  <soap:Header/>
  <soap:Body>
    <urn:SendMessageResponse>
      <urn:DocumentReferenceNumber>34838cba-b51d-4215-b98f-0af0a34903e6</urn:DocumentReferenceNumber>
    </urn:SendMessageResponse>
  </soap:Body>
</soap:Envelope>
```

5.5.3 Process message request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01"
xmlns:urn2="urn:fi:Datahub:mif:common:HDR_Header:elements:v1" xmlns:urn3="urn:fi:Datahub:mif:common:PEC_ProcessEnergyContext:elements:v1"
xmlns:urn29="urn:fi:Datahub:mif:query:F17_PartyInfoRequest:v1" xmlns:urn30="urn:fi:Datahub:mif:query:F17_PartyInfoRequest:elements:v1">
 <soap:Header/>
 <soap:Body>
  <urn:ProcessMessageRequest>
   <urn:MessageContainer>
    <urn:Payload>
     <urn29:PartyInfoRequestMessage>
      <urn29:PartyInfoRequest>
       <urn29:Header>
        <urn2:Identification>345345345345345</urn2:Identification>
        <urn2:DocumentType>F17</urn2:DocumentType>
        <urn2:Creation>2018-01-01T10:10:00</urn2:Creation>
        <urn2:PhysicalSenderEnergyParty>
         <urn2:Identification schemeAgencyIdentifier="9">8700000000001</urn2:Identification>
        </urn2:PhysicalSenderEnergyParty>
        <urn2:JuridicalSenderEnergyParty>
         <urn2:Identification schemeAgencyIdentifier="9">8700000000001</urn2:Identification>
        </urn2:JuridicalSenderEnergyParty>
        <urn2:JuridicalRecipientEnergyParty>
         <urn2:Identification schemeAgencyIdentifier="9">8700000000002</urn2:Identification>
        </urn2:JuridicalRecipientEnergyParty>
        <urn2:PhysicalRecipientEnergyParty>
         <urn2:Identification schemeAgencyIdentifier="9">8700000000002</urn2:Identification>
        </urn2:PhysicalRecipientEnergyParty>
        <urn2:SenderRoutingInformation>rtyrtyrty</urn2:SenderRoutingInformation>
       </urn29:Header>
       <urn29:ProcessEnergyContext>
        <urn3:EnergyBusinessProcess>DH-921-1</urn3:EnergyBusinessProcess>
        <urn3:EnergyBusinessProcessRole>DDQ</urn3:EnergyBusinessProcessRole>
        <urn3:EnergyIndustryClassification>23</urn3:EnergyIndustryClassification>
       </urn29:ProcessEnergyContext>
       <urn29:Transaction>
        <urn30:PartyIdentification schemeAgencyIdentifier="9">8700000000001</urn30:PartyIdentification>
       </urn29:Transaction>
      </urn29:PartyInfoRequest>
     </urn29:PartyInfoRequestMessage>
    </urn:Payload>
   </urn:MessageContainer>
  </urn:ProcessMessageRequest>
 </soap:Body>
</soap:Envelope>
```

5.5.4 Process message response

Example of a response message having an Acknowledgement as answer:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01"
xmlns:urn2="urn:fi:Datahub:mif:common:HDR_Header:elements:v1" xmlns:urn3="urn:fi:Datahub:mif:common:PEC_ProcessEnergyContext:elements:v1"
xmlns:urn13="urn:fi:Datahub:mif:common:ACK_Acknowledgement:v1" xmlns:urn14="urn:fi:Datahub:mif:common:ACK_Acknowledgement:elements:v1">
  <soap:Header/>
  <soap:Body>
   <urn:ProcessMessageResponse>
```

```
      <urn:MessageContainer>
        <urn:DocumentReferenceNumber>12a10e73-dd65-4787-9ad7-a40aa533f225</urn:DocumentReferenceNumber>
        <urn:Payload>
          <urn13:AcknowledgementMessage>
            <urn13:Acknowledgement>
                …
            </urn13:Acknowledgement>
          </urn13:AcknowledgementMessage>
        </urn:Payload>
      </urn:MessageContainer>
    </urn:ProcessMessageResponse>
  </soap:Body>
</soap:Envelope>
```

5.5.5 Peek message request

Without domain

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
  <soap:Header/>
  <soap:Body>
    <urn:PeekMessageRequest/>
  </soap:Body>
</soap:Envelope>
```

With one domain

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
  <soap:Header/>
  <soap:Body>
    <urn:PeekMessageRequest>
      <urn:MessageDomains>
        <urn:MessageDomain>BB01</urn:MessageDomain>
      </urn:MessageDomains>
    </urn:PeekMessageRequest>
  </soap:Body>
</soap:Envelope>
```

With multiple domains

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
  <soap:Header/>
  <soap:Body>
    <urn:PeekMessageRequest>
      <urn:MessageDomains>
        <urn:MessageDomain>BB01</urn:MessageDomain>
        <urn:MessageDomain>BB02</urn:MessageDomain>
      </urn:MessageDomains>
    </urn:PeekMessageRequest>
  </soap:Body>
</soap:Envelope>
```

5.5.6 Peek message response

5.5.6.1 Peek response with no available message in the message queue

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
  <env:Header/>
  <env:Body>
    <urn:PeekMessageResponse/>
  </env:Body>
</env:Envelope>
```

5.5.6.2 Peek response with available message in the message queue

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01"
xmlns:urn1="urn:fi:Datahub:mif:masterdata:F01_MasterDataCustomerEvent:v1" xmlns:urn2="urn:fi:Datahub:mif:common:HDR_Header:elements:v1"
xmlns:urn3="urn:fi:Datahub:mif:common:PEC_ProcessEnergyContext:elements:v1"
xmlns:urn4="urn:fi:Datahub:mif:masterdata:F01_MasterDataCustomerEvent:elements:v1"
   <soap:Header/>
   <soap:Body>
      <urn:PeekMessageResponse>
         <urn:MessageContainer>
            <urn:DocumentReferenceNumber>?</urn:DocumentReferenceNumber>
            <urn:Payload>
               <urn1:MasterDataCustomerEventMessage>
                  <urn1:MasterDataCustomerEvent>
                        …
                  </urn1:MasterDataCustomerEvent>
               </urn1:MasterDataCustomerEventMessage>
            </urn:Payload>
         </urn:MessageContainer>
      </urn:PeekMessageResponse>
   </soap:Body>
</soap:Envelope>
```

5.5.7 Dequeue message request

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
   <soap:Header/>
   <soap:Body>
      <urn:DequeueMessageRequest>
         <urn:DocumentReferenceNumber>8f3d3929-2dcd-475a-8824-5a566f128bd7</urn:DocumentReferenceNumber>
      </urn:DequeueMessageRequest>
   </soap:Body>
</soap:Envelope>
```

5.5.8 Dequeue message response

Accepted:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01">
   <soap:Header/>
   <soap:Body>
      <urn:DequeueMessageResponse/>
   </soap:Body>
</soap:Envelope>
```

A XML Signature Examples

This appendix provides several examples of XML Signatures, where each significant section is highlighted. See section 2.6 for more information.

Note that in the examples some of the digests have been shortened for readability purposes.

The following examples are provided:

#	Example description	Section reference
	Using BinarySecurityToken that contains the certificate data required for validating the XML Signature.	A.1
	Using a WSU Timestamp element to indicate when the Signature lifetime (when it was created and when it is no longer valid).	A.2
	Using XPath to filter the referenced signed scope to enable excluding SOAP elements from the Signature that may be changed. Only use this pattern when needed and in agreement with Fingrid.	**Error! Reference source not found.**
	An invalid example of using XPath to reference the signed scope but not providing a URI that references the main nodes based on their WSU Id. This example will be rejected by Datahub.	**Error! Reference source not found.**

Most of these examples are based on signing of the whole Body element. Signing of only the Business message is also allowed as it has an optional wsu:id attribute. Deeper elements don’t have this wsu:id and will give schema validation errors. The XML fragment below shows the syntax of these two variants:

```
 …
 <soapenv:Body>
  <urn:SendMessageRequest>
    <urn:MessageContainer>
     <urn:Payload>
       <urn1:MasterDataCustomerEventMessage xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-df678ba4-ff74-4864-9dec-86921f443bc1">
           <urn1:MasterDataCustomerEvent>
…
```

```
 …
 <soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-da76e06b-205f-4024-a37b-e9551a5e1ce0">
   <urn:SendMessageRequest>
     <urn:MessageContainer>
       <urn:Payload>
  <urn1:MasterDataCustomerEventMessage>
           <urn1:MasterDataCustomerEvent>
…
```

A.1 Example using BinarySecurityToken

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01" xmlns:urn1="urn:cms:b2b:message:v01:v1">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" soapenv:mustUnderstand="true">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="X509-d63b7c20-9243-40a9-9328-e2682168f8ed">MIIFYzCCA0ugAwIBAgIQQs3...</wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="SIG-b5d59b7b-64e2-48a4-a584-e682655de1ae">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
          <ds:Reference URI="#id-df678ba4-ff74-4864-9dec-86921f443bc1">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <InclusiveNamespaces xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="urn urn1"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>zLXPEmfLjw4mA5BVjwOLPy09a36ZGhZZ0VFJ3p16yI8=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>ud9VRpSwW9SeRhAQNDm7zRM...</ds:SignatureValue>
        <ds:KeyInfo Id="KI-1062e5f6-18e4-4aa5-89e3-4652d67488ff">
          <wsse:SecurityTokenReference wsu:Id="STR-b7d5914b-73a2-4594-b8b7-61fa66298213">
            <wsse:Reference URI="#X509-d63b7c20-9243-40a9-9328-e2682168f8ed" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-df678ba4-ff74-4864-9dec-86921f443bc1">
    <urn:SendMessageRequest>
      <urn:MessageContainer>
        <urn:Payload>
          <urn1:CreateMeteringPointRequest>
            <urn1:MessageHeader>
            …
            </urn1:MessageHeader>
            <urn1:Process>
            …
            </urn1:Process>
            <urn1:Transaction>
            …
            </urn1:Transaction>
          </urn1:CreateMeteringPointRequest>
        </urn:Payload>
      </urn:MessageContainer>
    </urn:SendMessageRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A.2 Example using Timestamp

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:urn="urn:cms:b2b:v01" xmlns:urn1="urn:cms:b2b:message:v01:v1">
  <soapenv:Header>
   <wsse:Security soapenv:mustUnderstand="true" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
    <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="X509-11628821-2541-4f78-ab43-bbf5c1967146">MIIFYzCCA0ugAwIBAgIQQs3...</wsse:BinarySecurityToken>
    <ds:Signature Id="SIG-3923a2ea-98ea-4fa3-b6da-098c68e059eb" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
     <ds:SignedInfo>
       <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
       <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
       <ds:Reference URI="#id-b3a822c1-fb46-4e20-9e40-4429d9d43b84">
        <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>DZVMxMPxsFdZsWIBoiQJNGoaNKkZiWH4TjQHQ8d2JBg=</ds:DigestValue>
       </ds:Reference>
       <ds:Reference URI="#TS-b5ebb6fd-d720-4378-b35c-4c4d4eede148">
        <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>TjZCNGSdLT8V2KtynBx7lITUlkHu8lahahFFwoB5Kec=</ds:DigestValue>
       </ds:Reference>
     </ds:SignedInfo>
     <ds:SignatureValue>kIKM2Y0KpvSQKIrTA5KG3c1...</ds:SignatureValue>
     <ds:KeyInfo Id="KI-b9eb66bc-b1cc-4a2f-a0df-069970a6ece6">
       <wsse:SecurityTokenReference wsu:Id="STR-2ecdcd45-e3fb-4854-91ef-5df6b6783fea">
        <wsse:Reference URI="#X509-11628821-2541-4f78-ab43-bbf5c1967146" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
       </wsse:SecurityTokenReference>
     </ds:KeyInfo>
    </ds:Signature>
    <wsu:Timestamp wsu:Id="TS-b5ebb6fd-d720-4378-b35c-4c4d4eede148">
     <wsu:Created>2020-02-14T11:14:21.556Z</wsu:Created>
     <wsu:Expires>2020-02-14T11:19:21.556Z</wsu:Expires>
    </wsu:Timestamp>
   </wsse:Security>
  </soapenv:Header>
  <soapenv:Body wsu:Id="id-b3a822c1-fb46-4e20-9e40-4429d9d43b84" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <urn:SendMessageRequest>
    <urn:MessageContainer>
     <urn:Payload>
       <urn1:CreateMeteringPointRequest>
        <urn1:MessageHeader>
          …
        </urn1:MessageHeader>
        <urn1:Process>
          …
        </urn1:Process>
        <urn1:Transaction>
          …
        </urn1:Transaction>
        </urn1:CreateMeteringPointRequest>
     </urn:Payload>
    </urn:MessageContainer>
   </urn:SendMessageRequest>
  </soapenv:Body>
</soapenv:Envelope>
```