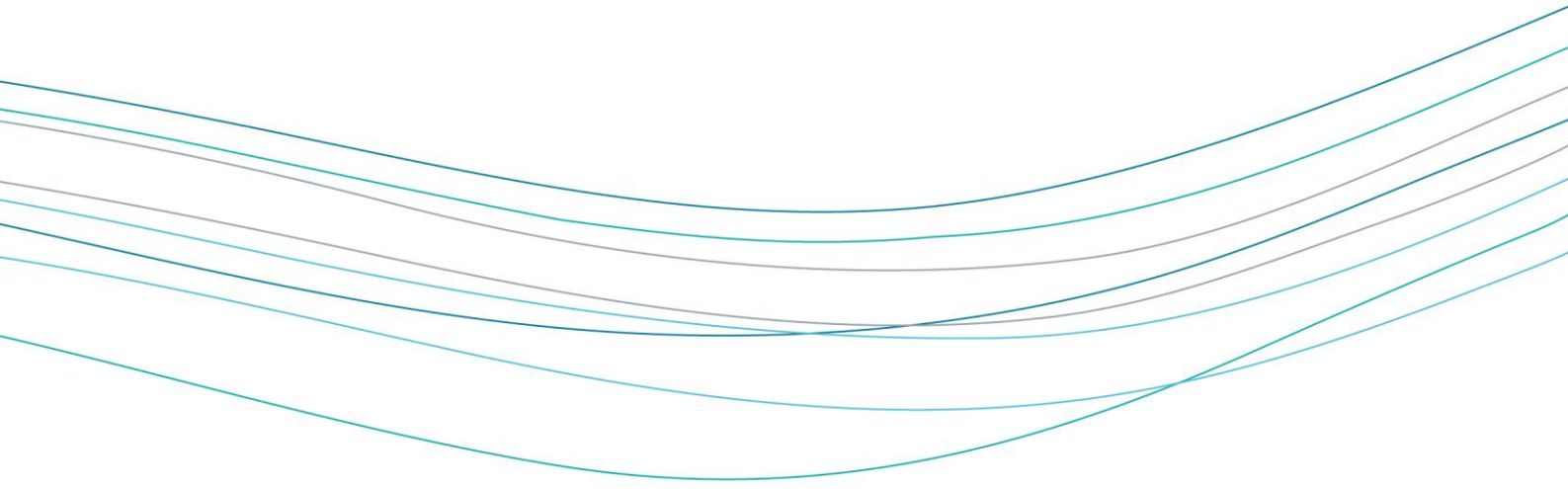
Bilateral Hub External Interface Specification

Version 0.3 Draft



Version history

Version	Update	Date
0.1	Initial draft	30/11/2020
0.2	Updated draft	01/12/2020
0.3	Includes idempotent transaction submission for REST messages	10/01/2021

Distribution

This document has been distributed to the following people:

Name	Purpose	Date of Issue	Version
Pathfinder Group	To enable pathfinder organisations to start development work and identify any gaps in the document.	01/12/2020	0.1

Attached Documents

Document Title	Filename	Date	Version
None			

Glossary

Term / Abbreviation / Acronym	Meaning
MOSL	Market Operator Services Limited
CMOS	Central Market Operating System
CSD	Code Subsidiary Document
TP	Trading Party
JSON	JavaScript Object Notation
CSD 0601	Bilateral Data Catalogue
XML	Extensible Markup Language
WSDL	Web Service Definition Language
GUID	Globally Unique Identifier

Contents

About this document

Purpose of the External Interface Specification Document

This document contains the external interface specification for the HVI messaging channel provided by the Bilateral Hub. This document details the interfaces as specified in CSD 0400 and CSD 0401 and should be read in conjunction with CSD 0601 – Bilateral Data Catalogue.

Scope

This document describes how messages are exchanged with the Bilateral Hub and the underlying transport and security mechanism. It defines the transport, authentication and syntax checks that will be applied to each inbound message, but does not include the transaction specific content validation rules that are defined in the addendum to CSD 0601.

Target Audience

The target audiences for this document are the technical designers and implementation teams that will implement the Trading Party interfaces to interface with the Bilateral Hub.

Overview

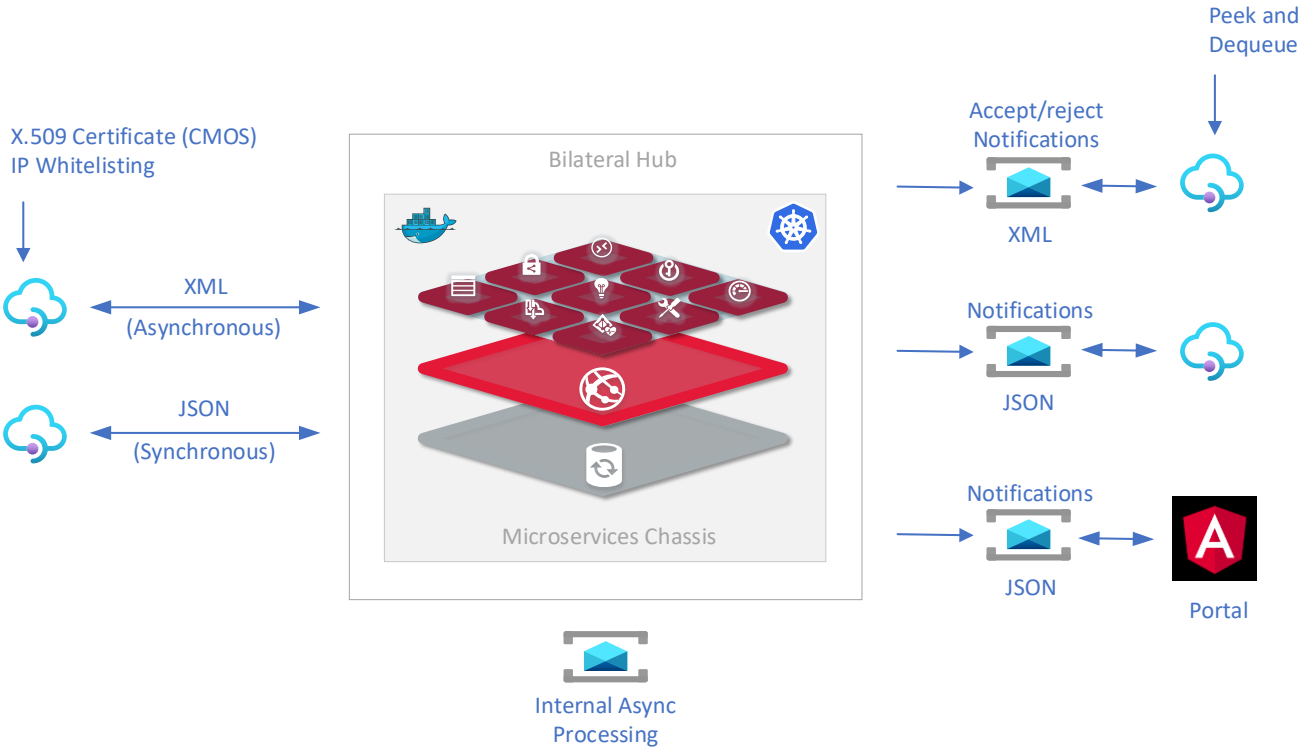
Messaging

Connectivity to the Bilateral Hub can be achieved by using either XML or JSON web services. Trading Parties wishing to adopt the HVI for some or all of the transactions defined in CSD 0601 will select the format that is most appropriate for their specific implementation requirements. A Trading Party will only be able to use one format or the other.

The XML interface will use asynchronous processing as defined in CSD 0401. The hub will return a synchronous response once authentication and initial validation is complete with the processing of the transaction occurring in the background. A notification will be generated once processing is complete to inform the originator of the outcome.

The JSON interface will use synchronous processing where all validation will be undertaken in real-time and the notification will be delivered in the synchronous response.

Trading Parties will connect to the Bilateral Hub to send messages containing transaction information and download both solicited and unsolicited notifications. The architecture is shown in the diagram below.



Bilateral Hub Messaging Architecture

The HVI interface will be protected using X.509 certificates and IP whitelisting in the same way that CMOS is secured. The X.509 certificate will be used to identify the Trading Party and this will be validated against the transaction header and the message will be rejected if the details do not match.

Trading Parties will send transactions and receive notifications using their preferred format. It will be the responsibility of the hub to manage the translation between XML and JSON. For example, a retailer may send a transaction to the hub in XML format and the wholesaler may receive the notification in JSON format and vice-versa.

Operations

The interface will support three generic operations:

- Send - to send transactions to the Bilateral Hub. The message and the transaction inside the message are logged, authenticated and authorised and checked for message integrity.

  For JSON transactions the content of the transaction is also validated against transaction specific business rules. For XML messages the transaction specific validation rules are processed asynchronously and notified to the initiating system using the peek and dequeue mechanism.

  Both JSON and XML interfaces will return a synchronous response to the initiating system detailing the success or failure of the message.

- Peek - is used by the Trading Party to retrieve the content of the first message available to it. The Trading Party can then process the message asynchronously and once successful can dequeue it.

- De-queue - the Trading Party informs the Bilateral Hub that the message has been processed. If a message is not De-queued, any subsequent Peek message will return the same message. Note that only the last peeked message can be dequeued. Any attempt to dequeue any other message will result in failure.

The Send/Peek/dequeue protocol ensures transactional integrity between the Trading Party system and the Bilateral Hub.

Endpoints

For each Bilateral Hub environment, the XML interface will be implemented as a single web service using a SOAP envelope. The WSDL for the XML interface is provided alongside the XSD. For the JSON interface there will be 3 endpoints to map to each of the operations.

Message Structure

The following sections define the message structure used to communicate with the Bilateral Hub. The structure and contents of each message will be the same for both XML and JSON interfaces unless explicitly indicated otherwise. Each message sent to the hub will receive a synchronous response or an HTTP error. HTTP status codes that may be returned by the hub are described in the section HTTP Level Validation.

Message Container

All transactions sent to the Bilateral Hub will be enclosed within a message container with a unique document reference as defined in CSD 0401. The structure of the message container is shown in the table below.

Name	Mand / Opt	Type	Length	Description
MessageContainer	M			
DocumentReferenceNumber	M	String	38	The DocumentReferenceNumber is used to identify the data transfer of a message from either the Trading Party system or the Bilateral Hub. The DocumentReferenceNumber must be unique over time and across trading parties. The recommended algorithm for creating this ID is the GUID algorithm.
DocumentTransactionType	M	Enumeration		The DocumentTransactionType refers to the role of the initiating party in the market for this transaction. The permitted values are shown in the table Document Transaction Types.
DataTransactionFormat	M	Enumeration		The DataTransactionFormat defines the enclosed message format within the payload and can either be 'XML' or 'JSON'.
Payload	M			Contains the actual business transaction in the format as specified in the DataTransactionFormat. The business transaction must be structured in accordance with CSD 0601.

Message Container

Document Transaction Types

The table below shows the 3 document transaction types that can be submitted in the message container. This will be validated against the role of the Trading Party submitting a transaction.

DocumentTransactionType	Description
WholesalerTransaction	Document contains a transaction of type *.W
RetailerTransaction	Document contains a transaction of type *.R
MarketOperatorTransaction	Document contains a transaction of type *.M

Document Transaction Types

SendMessage

A SendMessageRequest uses a MessageContainer to send a single transaction to the Bilateral Hub. The transaction is contained within the payload.

Name	Mand / Opt	Type	Length	Description
SendMessageRequest	M			
MessageContainer	M			
DocumentReferenceNumber	M	String	38	The DocumentReferenceNumber is used to identify the data transfer of a message from either the Trading Party system or the Bilateral Hub.
DocumentTransactionType	M	Enumeration		The DocumentTransactionType refers to the role of the initiating party in the market for this transaction. The permitted values are shown in the table Document Transaction Types.
DataTransactionFormat	M	Enumeration		The DataTransactionFormat defines the enclosed message format within the payload and can either be 'XML' or 'JSON'.
Payload	M		10Mb	Contains the actual business transaction in the format as specified in the DataTransactionFormat. The business transaction must be structured in accordance with CSD 0601.

SendMessageRequest

The following rules are enforced:

- The DocumentReferenceNumber has not been used before
- The Payload maximum size is less than 10Mb
- The Payload is in line with the mentioned DocumentTransactionType and DataTransactionFormat
- The Payload is syntactically correct
- The user, identified by the X.509 certificate, is authorised to send in the document type
- The DataTransactionReferenceNumber in the payload has not been used before
- The OriginatorsReference in the payload has not been used before

Example of a SendMessageRequest for a T201.W_AcceptServiceRequest

```
<SendMessageRequest>
        <MessageContainer>
                <DocumentReferenceNumber />
                <DocumentTransactionType />
                <DataTransactionFormat />
                <Payload>
                        <Transaction>
                                <T201.W_AcceptServiceRequest>
                                </T201.W_AcceptServiceRequest>
                        </Transaction>
                </Payload>
        </MessageContainer>
</SendMessageRequest>
```

SendMessageRequest Example

SendMessageResponse

The SendMessageResponse is returned in the synchronous response to the SendMessageRequest and may contain either the DocumentReferenceNumber if the synchronous element of the XML message was successful or the rejections if the XML message failed authentication and initial validation.

For JSON messages the SendMessageResponse will contain either a rejection if the message cannot be processed or a MessageContainer with either the T209.M or T219.M notifications.

The SendMessageResponse will only contain one of the 3 element types.

Name	Mand / Opt	Type	Length	Description
SendMessageResponse	M			
DocumentReferenceNumber	O	String	38	The DocumentReferenceNumber provided in the SendMessageRequest if the transaction is successful and the DataTransactionFormat is XML.
Rejections	O	Rejection		Populated where there are errors and the DataTransactionFormat is XML. The structure of a rejection is shown in the table Rejection Structure.
MessageContainer	O			Populated when the DataTransactionFormat is JSON with either a T209.M rejection notification or a T219.M accept notification.

SendMessageResponse

Message Rejection Structure

Name	Mand / Opt	Type	Length	Description
ErrorCode	M	String	20	A validation error code from the set of error codes defined in the section
ErrorDetails	O	String	255	A description of the error to assist the user with identifying the root cause

Rejection Structure

Example of a JSON SendMessageResponse for a T209.M_NotifyTransactionRejected notification

```
{
  "SendMessageResponse": {
    "MessageContainer": {
      "Payload": {
        "Transaction": { "T209.M_NotifyTransactionRejected": "---T209.M Header and Payload---" }
      }
    }
  }
}
```

SendMessageResponse JSON

Example of an XML SendMessageResponse for an accepted transaction

```
<SendMessageResponse>
        <DocumentReferenceNumber />
</SendMessageResponse>
```

SendMessageResponse XML

PeekMessage

The HVI system uses PeekMessage to retrieve a notification from the Bilateral Hub. It will return the top most message (FIFO) on the logical queue that has not been marked as 'Dequeued'.

Please note that PeekMessage returns a notification that can be processed by the caller of PeekMessage, without dequeuing this business message first.

It is the responsibility of the Trading Party to regularly retrieve, process and dequeue messages. The Bilateral Hub will continue processing and preparing additional messages independent of the Trading Party retrieving messages. Messages will be delivered in the order that the Bilateral Hub has created the notifications.

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

Name	Mand / Opt	Type	Length	Description
PeekMessageRequest	M			

Example of a PeekMessageRequest

```
<PeekMessageRequest/>
```

PeekMessageRequest Example

PeekMessageResponse

Name	Mand / Opt	Type	Length	Description
PeekMessageResponse	M			
MessageContainer	O			The MessageContainer will be present if there is a message queued on the interface.
DocumentReferenceNumber	M	String	38	The DocumentReferenceNumber will be generated by the Bilateral Hub.
DocumentTransactionType	M	Enumeration		The DocumentTransactionType refers to the role of the initiating party in the market for this transaction. The permitted values are shown in the table Document Transaction Types.
DataTransactionFormat	M	Enumeration		The DataTransactionFormat defines the enclosed message. This will be set by the Bilateral Hub using the Trading Party preferred notification format.
Payload	M			

Please note that on subsequent PeekMessage requests that would return the same message if the DocumentReferenceNumber is not updated. The initial PeekMessage request is interpreted as technical not succeeded, hence the original DocumentReferenceNumber is used in the response. This DocumentReferenceNumber is generated once by the Bilateral Hub.

Example of a PeekMessageResponse for an accepted transaction

```
<PeekMessageResponse>
        <MessageContainer>
                <DocumentReferenceNumber />
                <DocumentTransactionType />
                <DataTransactionFormat />
                <Payload>
                        <Transaction>
                                <T201.M_NotifyServiceRequestAccepted>
                                </T201.M_NotifyServiceRequestAccepted>
                        </Transaction>
                </Payload>
        </MessageContainer>
</PeekMessageResponse>
```

PeekMessageRequest Example

DequeueMessage

The Trading Party system uses DequeueMessage to indicate to the Bilateral Hub it has handled a message it retrieved from the hub using PeekMessage and has been able to extract the content in order to use the DocumentReferenceNumber of the message to dequeue. This message will be removed from the logical queue for this Trading Party. Note that when this is done the next message comes available on the logical queue for peeking.

Note that attempting to dequeue a message that has been dequeued or is not the last message peeked with return an error.

Name	Mand / Opt	Type	Length	Description
DequeueMessageRequest	M			
DocumentReferenceNumber	M	String	38	The DocumentReferenceNumber from the last PeekMessageResponse.

DequeueMessageResponse

Name	Mand / Opt	Type	Length	Description
DequeueMessageResponse	M			
DocumentReferenceNumber	O	String	38	The reference from the PeekMessageResponse if the message was successfully dequeued
Rejection	O	Rejection		The error details if the dequeue failed

Message Validation

Messages will be validated at 4 levels. For both XML and JSON validation transport, authentication and message validation will occur synchronously. The hub will include a rejection code in the response if validation fails at any of these stages.

If the validation is successful, the content of the transaction contained within the message will be processed asynchronously for XML requests and will generate a T209.M or T219.M that can be peeked and subsequently dequeued. For JSON requests it will be validated synchronously and return the T209.M or the T219.M in the synchronous response.

Transport Level Responses

Transport level failures will be generated in the form of HTTP Status Codes as defined in the table below. Note that any message that receives a transport level failure will not be stored or processed by the Bilateral Hub.

Error Code	Type	Meaning
200	Success	OK – The request has been accepted for processing. Any subsequent errors found during the processing of the message will be returned in the synchronous response.
403	Client	Forbidden – The security certificate supplied is not known to the Bilateral Hub or the IP address is not whitelisted on the hub
404	Client	Not Found – The specified endpoint does not exist
413	Client	Request Entity Too Large – The message payload exceeds 10 Mb
429	Client	Too Many Requests – The client has exceeded the maximum rate of transaction submissions
500	System	Internal Server Error – An unexpected failure in the system
503	System	Service Unavailable – The service is undergoing maintenance

Authentication and Authorisation

The X.509 certificate will be authenticated against the Trading Party and validation will check whether the Trading Party can submit the transaction contained in the payload. The errors that will be returned by the Bilateral Hub are defined in the table below.

Error Code	Description
AUTH-0001	User is blocked or inactive
AUTH-0002	User does not have permissions to submit the transaction
AUTH-0003	Trading Party Identifier not recognised
AUTH-0004	Invalid certificate type
AUTH-0005	Back-end timeout

Message Validation

Messages that do not have unique identifiers or do not comply with the transaction definitions defined in CSD 0601 will be logged but not processed. The table below shows the list of synchronous error codes that will be returned to the client in the event of a validation failure.

Error Code	Description
MESS-0001	Schema validation failure
MESS-0002	Unrecognised transaction type
MESS-0003	DocumentReferenceNumber must be unique
MESS-0004	DataTransactionReferenceNumber must be unique
MESS-0005	OriginatorsReference must be unique
MESS-0006	The DocumentReferenceNumber is not recognised or is not the last messaged Peeked
MESS-0007	Back-end timeout

When the client fails to receive a synchronous response as the result of either a network, client or hub failure, the message should be resubmitted. If the Bilateral Hub had not receipted and processed the message then the message will be processed normally and the client will receive the appropriate response.

If the message had been receipted by the Bilateral Hub (the DocumentReferenceNumber and DataTransactionReferenceNumber have been used in a previous message), the Bilateral Hub will respond with a duplicate message error, but the response will contain the original synchronous response.

Content Validation

Content validation rules are not in the scope of this document. They are provided as an addendum to CSD 0601.