# Simpl.ePRIOR / MercuriusService - Technical Guide

Ver. 1.3

Version History

|  |  |  |
| --- | --- | --- |
| Version | Changes | Author |
| **1.0** | **Initial version** | **DWL** |
| **1.1** | **Document types added** | **DWL** |
| **1.2** | **Add outbound invoicing** | **DWL** |
| **1.3** | **Add outbound invoicing and CustomizationID’s/Profiles formats** | **DWL** |

Table of Contents

[Simpl.ePRIOR / MercuriusService - Technical Guide 1](#_Toc530474494)

[Introduction 3](#_Toc530474495)

[Context 4](#_Toc530474496)

[The Global Picture 5](#_Toc530474497)

[WSDL 6](#_Toc530474498)

[Submit Inbox Request 6](#_Toc530474499)

[Request 6](#_Toc530474500)

[Response 7](#_Toc530474501)

[Retrieve Document 7](#_Toc530474502)

[Request 7](#_Toc530474503)

[Response 8](#_Toc530474504)

[Mark Document 9](#_Toc530474505)

[Request 9](#_Toc530474506)

[Response 9](#_Toc530474507)

[Submit Document 10](#_Toc530474508)

[Request 10](#_Toc530474509)

[Response 11](#_Toc530474510)

[Soap with attachment 11](#_Toc530474511)

[Test Plan (Mock service) 14](#_Toc530474512)

[Endpoint 14](#_Toc530474513)

[Mockservice (if started locally) 14](#_Toc530474514)

[Submit Inbox Request 14](#_Toc530474515)

[Request 14](#_Toc530474516)

[Response 15](#_Toc530474517)

[Available Document Types supported yet 16](#_Toc530474518)

[Announced for next releases 16](#_Toc530474519)

[Retrieve Document 16](#_Toc530474520)

[Request 16](#_Toc530474521)

[Response 17](#_Toc530474522)

[Mark Document 17](#_Toc530474523)

[Request 17](#_Toc530474524)

[Response 18](#_Toc530474525)

[Submit Document 18](#_Toc530474526)

[Available Document Types supported yet 18](#_Toc530474527)

[Announced for next releases 18](#_Toc530474528)

[Request 19](#_Toc530474529)

[Response 20](#_Toc530474530)

[Connecting to Mercurius TEST 22](#_Toc530474531)

[Endpoint 22](#_Toc530474532)

[Mercurius TEST environment (FSB service at BOSA DT) 22](#_Toc530474533)

[Connecting to the Mercurius TEST environment - FAQ 25](#_Toc530474534)

[How send testfiles to Mercurius (XML – PEPPOL)? 25](#_Toc530474535)

[How can I verify my testfile has arrived on Mercurius TEST? 25](#_Toc530474536)

[How can I validate my XML file for PEPPOL compliancy? 25](#_Toc530474537)

[How get access to Mercurius ? 25](#_Toc530474538)

[Connecting to Mercurius PROD 26](#_Toc530474539)

[Endpoint 26](#_Toc530474540)

[Mercurius PROD environment (FSB service at BOSA DT) 26](#_Toc530474541)

[Security 26](#_Toc530474542)

[Limitations 27](#_Toc530474543)

[Mercurius Portal changes 28](#_Toc530474544)

[Other scenarios 30](#_Toc530474545)

[Inbound Invoicing 30](#_Toc530474546)

[Outbound Invoicing (target implementation date 31/12/2018) 31](#_Toc530474547)

[Concept 31](#_Toc530474548)

[Logic 31](#_Toc530474549)

[Formats allowed PEPPOL BIS v2: 32](#_Toc530474550)

[Formats allowed PEPPOL BIS v3 (available 31/3/2019): 32](#_Toc530474551)

[Impact on portal 33](#_Toc530474552)

[Information how SBDH is constructed 33](#_Toc530474553)

[Final remarks 33](#_Toc530474554)

[Outbound e-Ordering (Target implementation date 31/12/2018) 34](#_Toc530474555)

[Concept 34](#_Toc530474556)

[Logic 34](#_Toc530474557)

[Formats allowed PEPPOL BIS v2: 35](#_Toc530474558)

[Formats allowed PEPPOL BIS v3 (available 31/3/2019): 35](#_Toc530474559)

[Impact on portal 36](#_Toc530474560)

[Information how SBDH is constructed 36](#_Toc530474561)

[Final remarks 36](#_Toc530474562)

[PEPPOL document type 63A Invoice Response 37](#_Toc530474563)

[PEPPOL BIS v3 document types (Target implementation date 31/3/2019) 37](#_Toc530474564)

[PEPPOL BIS v2 mapping to PEPPOL BIS v3 37](#_Toc530474565)

# Introduction

This document describes the (new) ePrior service exposed by the Mercurius Backend to its consumers, called Simpl.ePRIOR. The audience of this document are the consumers of the Mercurius services, i.e. IT teams of the Belgian public sector, responsible for integrating their accounting/procurement/invoices-handling software to the Mercurius e-Mailroom, that concentrates and consolidates the e-documents exchanges for the Belgian public sector.

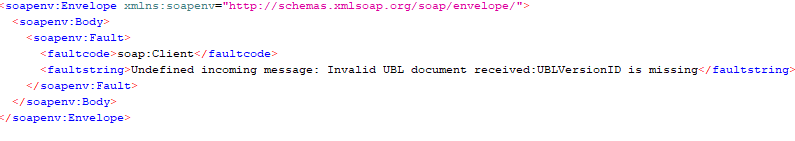
This document is not a stand-alone document. It goes along with (a) technical resources (WSDL, Test plan) needed by the IT teams to setup their connection and use the service as designed, and (b) additional information going deeper into the service usage pattern details: indeed, Simpl.ePRIOR is designed as a generic service, that can be used in various document exchange scenarios. The historical exchange scenario is b2g inbound e-invoicing, where (a) the supplier sends invoices and/or credit notes (with or without attached documents), and (b) the customer returns a business response (approval/rejection of the invoice). Simpl.ePRIOR is designed to support many additional exchange patterns, such as b2g e-ordering, g2b outbound e-invoicing, …. Furthermore, it is designed to support new e-document formats (due to the decision of Europe to impose a new format to all public-sector entities in Europe). However, the detailed explanation of these patterns clearly falls outside the scope of this document. This document provides only the generic foundation to using Simpl.ePRIOR to address these current and upcoming exchange patterns.

# Context

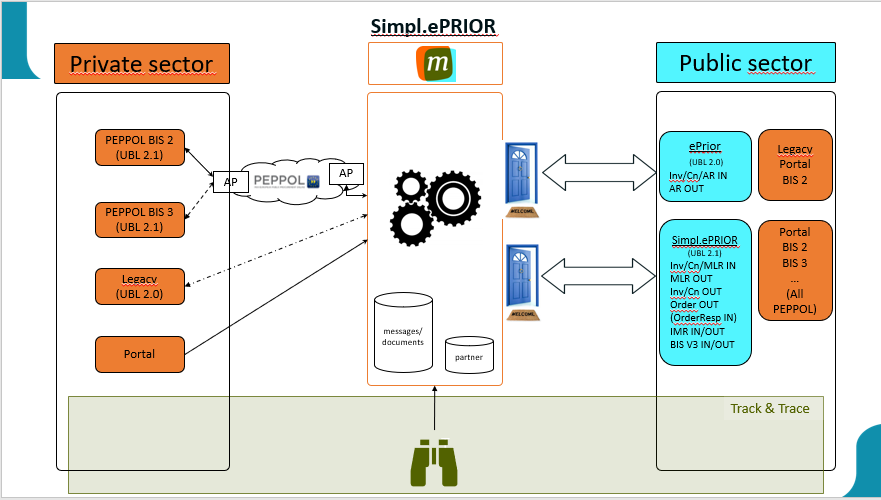
The current section consolidates the context elements that drive the implementation of Simpl.ePRIOR, and the essential features of Simpl.ePRIOR, that this context justifies. Next section provides detailed information about each operation composing the service.

1. Legacy context:
   * Currently, the Mercurius backend exposes a service called ePrior service, also called eInvoicingCustomerService, that only supports B2G inbound e-invoicing - more specifically, retrieve e-invoices/e-credit notes and attached documents, and send Application Responses back to the supplier. All documents are based on the CEN BII 2 WGA specifications and the UBL2.0 syntax.
   * For frictionless transition purposes, this legacy service keeps existing along with the Simpl.ePRIOR service, until it is no longer used by the existing consumers.
   * Consequently, Simpl.ePRIOR is exposed on a new service URL
   * Any existing customer of the current ePrior service, is able to use Simpl.ePRIOR.
2. PEPPOL alignment:
   * all documents handled by the Simpl.ePRIOR will strictly meet the PEPPOL specifications. For instance, UBL 2.1 syntax will be used.
   * Only explicitly motivated and documented deviations/transformations towards the PEPPOL specifications will be implemented. See next point.
3. Generic behaviour:
   * Simple.ePRIOR displays all received documents to the receiving entity, using one single presentation strategy: it returns the following metadata: From/To/DocType/DocId
   * Simpl.ePRIOR will validate all documents sent by the sending entity synchronously, using (a) xsd validation (based on UBL2.1 xsd untouched) and (b) executing applicable schematron validation (published by PEPPOL, docType-dependent). The b) part is executed based on the input value found in the fields CustomizationID and ProfileID. Simpl.ePRIOR will deduce the applicable schematron based on the value found in these 2 fields. If the values found do not enable Simpl.ePRIOR to successfully deduce the schematron to apply, the system will synchronously send a SOAP fault of type “soap:Client” (as described later in the document)
   * Simpl.ePRIOR transmits Attachments using SOAP with Attachments (multipart). This is the only transformation executed on all received documents, compared with their original PEPPOL specification.
4. Business validations (duplicate, etc.) performed by Mercurius, for instance upon reception of B2G e-invoices, e-credit notes and application Responses, fall out of the scope of the present document.
5. One or several integrators are used to connect all the service consumers to the Mercurius backend in a manageable way. The following list of considerations are specific to the FSB, which is the single system physically connected to the backend service.
   * The consumerID is passed as HTTP header by the FSB to the Simpl.ePRIOR service (same way as in the current ePrior service).
   * Both the consumerID and the FSBMessageID are logged in the message (in the Gateway OUT message status)
   * All Soap requests must be signed (WS-Security header), the same way it is done in the current ePrior service
6. Error Handling: whenever an error occurs while processing a request, Simpl.ePRIOR generates a generic SOAP 1.1 fault message containing the following elements:
   * faultCode:
     + "soap:Server" if there is an internal error in the server (similar to HTTP 500)
     + "soap:Client" if the error comes from the client (for example: invalid UBL message)
   * faultString: detailed message of the error

Example:



# The Global Picture



# WSDL

Compared with the Legacy ePrior service, Simpl.ePRIOR WSDL has been made generic. This considerably simplified the service. It only contains four operations:

1. **submitInboxRequest**: allows to list document in the inbox matching a given receiver and sender. This method does not retrieve any UBL document.
2. **retrieveDocument**: allows to retrieve any “supported” UBL document.
3. **markDocument**: allows to mark a document
4. **submitDocument**: allows to submit any “supported” UBL document.

The operations 1, 2 and 3 must be used to pick up the incoming documents, out of the inbox of one or several entities for which the service consumer is authorized. The classic usage pattern is to (a) get the list of new documents, and (b) for each new document, (i.) retrieve it and (ii.) upon successful completion of the retrieval, mark the document (so it is removed from the inbox).

The 4th operation must be used to send a document.

The next sections describe each of these operations in appropriate level of detail.

## Submit Inbox Request

### Request

The submitInboxRequest request takes 2 optional parameters that allow to filter the list of documents returned by the operation:

|  |  |
| --- | --- |
| **Name** | **How this value is retrieved in the existing E-Prior service?** |
| **receiverId** | //\*:RetrieveRequest/\*:ReceiverParty/\*:PartyIdentification/\*:ID/text() |
| **senderId** | //\*:RetrieveRequest/\*:SenderParty/\*:PartyIdentification/\*:ID/text() |

Remark:

Parameters are optional to specify. If none are present, then all documents for all receivers sent by all senders not yet marked with the same security lock (Common Name specified in certificate used in the connection) are listed. If one parameter is specified, then a selection is performed on the complete list. If both parameters are specified, then the selection is even more narrowed down. Some integrators require at least receiverId specified for security reasons (to be able to test receiverId against certificate information).

Example:

**SubmitInboxRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head=”<http://fsb.belgium.be/header>” xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<head:fsbHeader>

<head:messageId>9397cbe5-174f-479f-941d-2371129acb05</head:messageId>

</head:fsbHeader>

</soapenv:Header>

<soapenv:Body>

<v1:submitInboxRequest>

<v1:receiverId>?</v1:receiverId>

<v1:senderId>?</v1:senderId>

</v1:submitInboxRequest>

</soapenv:Body>

</soapenv:Envelope>

### Response

**SubmitInboxResponse**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00>

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:submitInboxRequestResponse>

<v1:documentGroup>

<!--Zero or more repetitions:-->

<v1:documentReference>

<v1:documentId>?</v1:documentId>

<v1:documentTypeCode>?</v1:documentTypeCode>

<v1:receiverId>?</v1:receiverId>

<v1:senderId>?</v1:senderId>

</v1:documentReference>

</v1:documentGroup>

</v1:submitInboxRequestResponse>

</soapenv:Body>

</soapenv:Envelope>

## Retrieve Document

### Request

The retrieveDocument request takes 4 parameters:

|  |  |
| --- | --- |
| **Name** | **How this value is retrieved in the existing E-Prior service?** |
| **receiverId** | //\*:RetrieveRequest/\*:ReceiverParty/\*:PartyIdentification/\*:ID/text() |
| **senderId** | //\*:RetrieveRequest/\*:SenderParty/\*:PartyIdentification/\*:ID/text() |
| **documentType** | //\*:RetrieveRequest/\*:DocumentReferenceRequest/\*:DocumentTypeCode/text() |
| **documentId** | //\*:RetrieveRequest/\*:DocumentReferenceRequest/\*:ID/text() |

Example:

**RetrieveDocumentRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head=”<http://fsb.belgium.be/header>” xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<head:fsbHeader>

<head:messageId>692da97e-e4a0-4ed0-b0bd-6e3f01e1e601</head:messageId>

</head:fsbHeader>

</soapenv:Header>

<soapenv:Body>

<v1:retrieveDocument>

<v1:receiverId>?</v1:receiverId>

<v1:senderId>?</v1:senderId>

<v1:documentType>?</v1:documentType>

<v1:documentId>?</v1:documentId>

</v1:retrieveDocument>

</soapenv:Body>

</soapenv:Envelope>

### Response

The retrieveDocument response returns the UBL document, as well as the received date (yyyy-MM-dd). Received date is the date the document was received on the Mercurius Platform.

**RetrieveDocumentResponse**

<soapenv:Envelope xmlns:soapenv=<http://schemas.xmlsoap.org/soap/envelope/> xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00>

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:retrieveDocumentResponse>

<v1:receivedDate>yyyy-MM-dd</v1:receivedDate>

<v1:document>

<!--Valid UBL document returned-->

</v1:document>

</v1:retrieveDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>

## Mark Document

### Request

The markDocument request takes the same parameters as the retrieve document.

Example:

**MarkDocumentRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head=”<http://fsb.belgium.be/header>” xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<head:fsbHeader>

<head:messageId>692da97e-e4a0-4ed0-b0bd-6e3f01e1e601</head:messageId>

</head:fsbHeader>

</soapenv:Header>

<soapenv:Body>

<v1:markDocument>

<v1:receiverId>?</v1:receiverId>

<v1:senderId>?</v1:senderId>

<v1:documentType>?</v1:documentType>

<v1:documentId>?</v1:documentId>

</v1:markDocument>

</soapenv:Body>

</soapenv:Envelope>

### Response

The response is exactly the same as the submit document response:

* In case of success, it returns a static SOAP response:

**MarkDocumentResponse**

<soapenv:Envelope xmlns:soapenv=”<http://schemas.xmlsoap.org/soap/envelope/>”

xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:markDocumentResponse>

<v1:ackIndicator>true</v1:ackIndicator>

</v1:markDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>

* In case of no success, it returns a static SOAP response with ackIndicator “false”.
* In case of failure or validation errors, it returns a SOAP fault with the detailed error message. The format of the SOAP fault is generic and is described earlier in the document.

## Submit Document

This single operation allows to submit any supported UBL document.

Each document will undergo the following validations (which are already done in the current ePrior service):

* Validation of the UBL document (using UBL 2.1 schematrons )

### Request

The request of submitDocument takes only one input which is a valid UBL document.

Example:

**SubmitDocumentRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head=”<http://fsb.belgium.be/header>” xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<head:fsbHeader>

<head:messageId>692da97e-e4a0-4ed0-b0bd-6e3f01e1e601</head:messageId>

</head:fsbHeader>

</soapenv:Header>

<soapenv:Body>

<v1:submitDocument>

<v1:document>

<!--You may enter ANY valid UBL document at this point-->

</v1:document>

</v1:submitDocument>

</soapenv:Body>

</soapenv:Envelope>

### Response

The response is similar to the response of the existing ePrior methods submitInvoice, submitOrder, etc.:

* In case of success, it returns a static SOAP response:

**SubmitDocumentResponse**

<soapenv:Envelope xmlns:soapenv=<http://schemas.xmlsoap.org/soap/envelope/>

xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:submitDocumentResponse>

<v1:ackIndicator>true</v1:ackIndicator>

</v1:submitDocumentResponse>

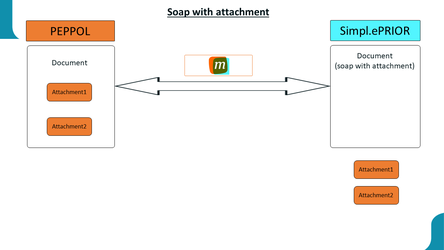
</soapenv:Body>

</soapenv:Envelope>

* In case of no success, it returns a static SOAP response with ackIndicator “false”.
* In case of failure or validation errors, it returns a SOAP fault with the detailed error message. The format of the SOAP fault is generic and is described earlier in the document.

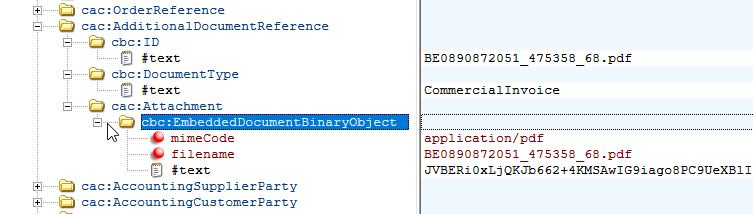
## Soap with attachment

Documents are received on the PEPPOL network with attachments included in the document. Mercurius is changing the presentation of the document to soap with attachment (meaning attachments are presented separated from the document body itself). The same logic is used when documents are sent to Mercurius (which will include the attachments in the document before putting the message on the PEPPOL network).



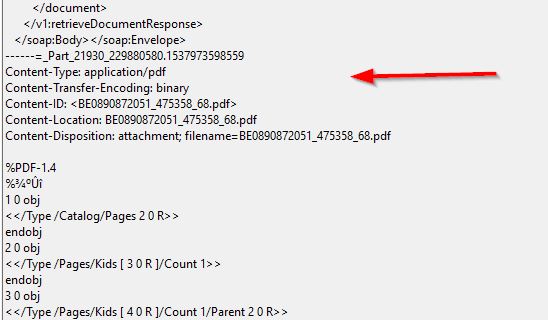
Example:

Source PEPPOL BIS v2 XML:



Result in Response:





Notice only the attachment itself is moved after the soap Envelope. All other elements concerning the attachment stay in place and filename is the link in case of multiple attachments.

# Test Plan (Mock service)

Along with the Simpl.ePRIOR WSDL, a test plan is provided. It contains a set of example request/responses illustrating the main flow supported by the service as well as the main exceptions. This test plan has a local mockup service to assist in the early development activities of the consumer IT team.

Following testplan and Mock service is only subject to provide examples on how requests and responses are constructed and can be used to determine which request is generating which response. Real business rules should be tested when connecting to the Mercurius test environment (see next chapter).

General remark: In all examples you can see a messageId in the <soapenv:Header> (format = uuid) to provide a unique identification on the message in the Mercurius/FSB logging.

## Endpoint

### Mockservice (if started locally)

<http://localhost:8088/mocksimplepriorSOAP>

current status:

* Test plan can be used locally as is (xml with test plan is available for import in SoapUI)
* Small set of examples are available for every operation
* Local Mock service can be used to test responses on customer developments
* More test requests can be added based upon provided examples
* We need to define soap with attachment in/out compared to PEPPOL BIS with attachments included

## Submit Inbox Request

### Request

#### 1.       Fakecustomer

This is an example to retrieve all documents based on 2 parameters: senderId en receiverId. Selection is possible by either one of them, both or none. The example is given for the customer “Fakecustomer” specifying a dummy value of ‘2222222222’ as the KBO number and sender “0200013901” as sample.

**SubmitInboxRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head="http://fsb.belgium.be/header" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<head:fsbHeader>

<head:messageId>9397cbe5-174f-479f-941d-2371129acb05</head:messageId>

</head:fsbHeader>

</soapenv:Header>

<soapenv:Body>

<v1:submitInboxRequest>

<v1:receiverId>2222222222</v1:receiverId>

<v1:senderId>0200013901</v1:senderId>

</v1:submitInboxRequest>

</soapenv:Body>

</soapenv:Envelope>

### Response

#### 1.       Response Fake customer list OK

The documentReference structure may be repeated several times depending on the number of documents ready to be retrieved. DocumentType may be different depending on the type of document available for download. The documentId will always have an ID in the format of uuid.

**SubmitInboxResponse**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:submitInboxRequestResponse>

<v1:documentGroup>

<!--Zero or more repetitions:-->

<documentReference>

<receiverId>2222222222</receiverId>

<senderId>0200013901</senderId>

<documentType>380</documentType>

<documentId>e878b54b-8a82-4bf3-8c8c-37eacdab2589</documentId>

</documentReference>

</v1:documentGroup>

</v1:submitInboxRequestResponse>

</soapenv:Body>

</soapenv:Envelope>

### Available Document Types supported yet

| **DocumentTypeName** | **UBL DocumentTypeCode** | **syntax (XSD)** | **semantic** |
| --- | --- | --- | --- |
| Invoice | 380 | [UBL-Invoice-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-Invoice-2.1.xsd) | [PEPPOL BIS 4A Invoice](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/InvoiceOnly4A) |
| Credit Note | 81 | [UBL-CreditNote-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-CreditNote-2.1.xsd) | [PEPPOL BIS 5A Billing](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/Billing5A) |
| Message Level Response | 301 | [UBL-ApplicationResponse-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-ApplicationResponse-2.1.xsd) | [PEPPOL BIS 36A Message Level Response](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/MessageLevelResponse36A) |

Remark: Notice that DocumentTypeCode depends on incoming elements e.g. for invoices InvoiceTypeCode is used (meaning 380 could be other value like 386 (prepayment invoice)). Same logic applies to other documents like credit note (CreditNoteTypeCode), … .

### Announced for next releases

| **DocumentTypeName** | **UBL DocumentTypeCode** | **syntax (XSD)** | **semantic** |
| --- | --- | --- | --- |
| Order Response | 231 | [UBL-OrderResponse-2.1.xsd](http://docs.oasis-open.org/ubl/cos1-UBL-2.1/xsd/maindoc/UBL-OrderResponse-2.1.xsd) | [PEPPOL BIS 28A Ordering](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/Ordering28A) |
| Invoice Response | 23 | [UBL-ApplicationResponse-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-ApplicationResponse-2.1.xsd) | [PEPPOL BIS 63A Invoice Response](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/InvoiceResponse63A) |

## Retrieve Document

### Request

#### 1.       Retrieve Invoice BIS 5A

The retrieve document operation has 4 mandatory parameters: receiverId, senderId, documentType and documentId. The content of these parameters can be determined by the SubmitInboxRequest operation.

**RetrieveDocumentRequest**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head="http://fsb.belgium.be/header" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header/>

<head:fsbHeader>

<head:messageId>9397cbe5-174f-479f-941d-2371129acb05</head:messageId>

</head:fsbHeader>

<soapenv:Body>

<v1:retrieveDocument>

<v1:receiverId>2222222222</v1:receiverId>

<v1:senderId>0200013901</v1:senderId>

<v1:documentType>380</v1:documentType>

<v1:documentId>e878b54b-8a82-4bf3-8c8c-37eacdab2589</v1:documentId>

</v1:retrieveDocument>

</soapenv:Body>

</soapenv:Envelope>

### Response

#### 1.       Response Invoice BIS 5A

The result of the retrieveDocument operation is the xml content of a document available on the Mercurius platform.

<RetrieveDocumentResponse.txt>

## Mark Document

### Request

#### 1.       Mark Invoice BIS 5A

The Mark document operation will eliminate the concerned document from the list retrieved by the SubmitInboxRequest operation. The mark document operation has 4 mandatory parameters: receiverId, senderId, documentType and documentId. The content of these parameters can be determined by the SubmitInboxRequest operation and is the same as used in the retrieve document operation.

**MarkDocument**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head="http://fsb.belgium.be/header" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header/>

<head:fsbHeader>

<head:messageId>9397cbe5-174f-479f-941d-2371129acb05</head:messageId>

</head:fsbHeader>

<soapenv:Body>

<v1:markDocument>

<v1:receiverId>2222222222</v1:receiverId>

<v1:senderId>0200013901</v1:senderId>

<v1:documentType>380</v1:documentType>

<v1:documentId>e878b54b-8a82-4bf3-8c8c-37eacdab2589</v1:documentId>

</v1:markDocument>

</soapenv:Body>

</soapenv:Envelope>

### Response

#### 1.       Response Invoice BIS 5A

The result of the Mark Document operation is a simple indication true/false to notify if the operation executed well or in error.

**MarkDocumentResponse**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:markDocumentResponse>

<v1:ackIndicator>true</v1:ackIndicator>

</v1:markDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>

## Submit Document

### Available Document Types supported yet

| **DocumentTypeName** | **syntax (XSD)** | **semantic** |
| --- | --- | --- |
| Message Level Response | [UBL-ApplicationResponse-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-ApplicationResponse-2.1.xsd) | [PEPPOL BIS 36A Message Level Response](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/MessageLevelResponse36A) |

### Announced for next releases

| **DocumentTypeName** | **syntax (XSD)** | **semantic** |
| --- | --- | --- |
| Invoice | [UBL-Invoice-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-Invoice-2.1.xsd) | [PEPPOL BIS 4A Invoice](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/InvoiceOnly4A) |
| Credit Note | [UBL-CreditNote-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-CreditNote-2.1.xsd) | [PEPPOL BIS 5A Billing](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/Billing5A) |
| Order | [UBL-Order-2.1.xsd](http://docs.oasis-open.org/ubl/cos1-UBL-2.1/xsd/maindoc/UBL-Order-2.1.xsd) | [PEPPOL BIS 28A Ordering](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/Ordering28A) |
|  |  | [PEPPOL BIS 3A Order only](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/OrderOnly3A) |
| Invoice Response | [UBL-ApplicationResponse-2.1.xsd](https://docs.oasis-open.org/ubl/prd3-UBL-2.1/xsd/maindoc/UBL-ApplicationResponse-2.1.xsd) | [PEPPOL BIS 63A Invoice Response](https://github.com/OpenPEPPOL/documentation/tree/master/PostAward/InvoiceResponse63A) |

Notice CustomizationID and ProfileID in the semantic documents specified above since schematrons validation will be based on those correct values identifying the document type.

### Request

#### 1.       Invoice BIS 5A

The Submit document operation can be used to send any document to Mercurius. This can be a new document (e.g. Outgoing invoice, outgoing order, …) or can be a response on a received document after processing (e.g. MLR (Message Level Response PEPPOL BIS 36A, IR (Invoice Response PEPPOL BIS 63A, …). The document itself will be contained in the soap message itself including attachments.

<SubmitDocument.txt>

#### 2.       Invalid soap request

This is an example of a malformed soap message being sent to Mercurius.

**SubmitDocumentRequestNoSoap**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:head="http://fsb.belgium.be/header" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header/>

<head:fsbHeader>

<head:messageId>9397cbe5-174f-479f-941d-2371129acb05</head:messageId>

</head:fsbHeader>

<soapenv:Body>

<v1:submitDocument>

<v1:document>

</v1:submitDocument>

</soapenv:Body>

</soapenv:Envelope>

#### 3.       Invalid doc. Type

This is an example of an unknown document type being sent to Mercurius. Mercurius currently support only a limited number of document types (even PEPPOL BIS is not fully supported for the moment) and is subject to change in the future.

<SubmitDocumentInvalidDocType.txt>

### Response

#### 1.       Response Invoice BIS 5A OK

The result of the Submit Document operation is a simple indication true/false to notify if the operation executed well or in error.

**SubmitDocumentResponse**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header>

<wsse:Security soap:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

<wsu:Timestamp wsu:Id="Timestamp-HvumButHp993jGDym7t89Q22" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

<wsu:Created>2018-10-03T10:07:55Z</wsu:Created>

<wsu:Expires>2018-10-03T10:12:55Z</wsu:Expires>

</wsu:Timestamp>

<!—More elements on security and signature:-->

</wsse:Security> </soapenv:Header>

<soapenv:Body>

<v1:submitDocumentResponse>

<v1:ackIndicator>true</v1:ackIndicator>

</v1:submitDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>

#### 2.       Invalid soap request response

The result of sending a malformed soap message to Mercurius results in an error message received back.

<submitDocumentResponseNoSoap.txt>

#### 3.       Soap fault – Invalid document

The result of sending an unknown document type (not yet supported?) to Mercurius.

**SubmitDocumentResponseInvalidDocType**

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:v1="http://fsb.belgium.be/einvoicing/simpleprior/v1\_00">

<soapenv:Header/>

<soapenv:Body>

<v1:submitDocumentResponse>

<faultcode>soap:Client</faultcode>

<faultstring>Undefined incoming message: Invalid UBL document received:UBLVersionID is missing</faultstring>

</v1:submitDocumentResponse>

</soapenv:Body>

</soapenv:Envelope>

# Connecting to Mercurius TEST

To be able to test the complete E2E flow of messages and business rules, Mercurius is equipped with a test environment identical to the production environment. This chapter contains instructions how to send test PEPPOL BIS messages to Mercurius TEST so they can be collected through the Simpl.ePRIOR MercuriusService.

## Endpoint

### Mercurius TEST environment (FSB service at BOSA DT)

see: [S331-MercuriusService (Simpl.ePRIOR)](http://dtservices.bosa.be/nl/services/fsb/web-services-families/eprocurementservices-s294/documentatie-eprocurementservices-s294#S331-MercuriusService%20(Simpl.ePRIOR))

wsdl: FSB version available at <https://fsb.services.int.belgium.be/eProcurementServices/PTS_MercuriusService?WSDL>

notice: previous URL is subject to change when connecting through one or more integrators. In this case the URL will be provided by the integrator.

remarks:

* Secure connection should be set up first and authorization to use the MercuriusService should be provided by FSB team BOSA.
* Current ePrior users are free to select the interface for retrieving documents on either interface. Notice however that the incoming legacy documents are only available on the old ePrior interface.
* Documents submitted through the portal (New Invoice Form) can be collected on either interface. Notice that format slightly differs (ePrior = UBL 2.0; Simpl.ePRIOR = UBL 2.1). Some content is generated automatically on Simpl.ePRIOR (e.g. BIC code in PaymentMeans since not mandatory in ePrior).

current status:

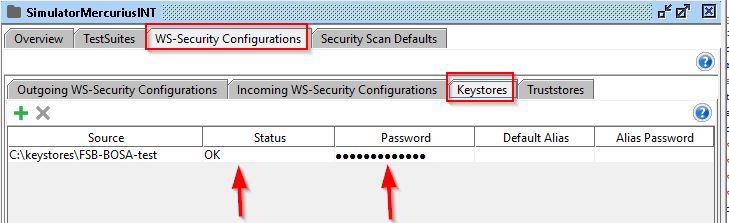
* PEPPOL IN connection fully available although testing still going on (same functionalities available as in old ePrior interface)
* PEPPOL OUT connection available (MLR and IR status response available after 9/12/2018)
* PEPPOL BIS V3 support not available yet (planned for Q1 2019: to be confirmed)
* PEPPOL BIS IR 63A OUT document type (planned: 31/12/2018)

SoapUI settings (for connecting to FSB – otherwise to be provided by integrator):

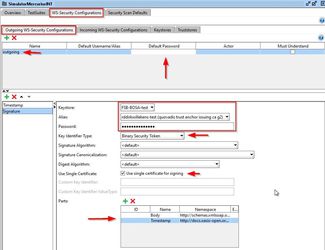
we assume:

1. you filled in the correct form for accessing the Simpl.ePRIOR service also called MercuriusService at [FSB eprocurementservices](http://dtservices.bosa.be/nl/services/fsb/web-services-families/eprocurementservices-s294/documentatie-eprocurementservices-s294)
2. the keystore has been prepared in FSB INT setup connection [connecting to FSB using a certificate](https://dtservices.bosa.be/nl/file/fedictfsbcertsandkeysmgmtpracticalguidelines22docx)

keystore settings:



signature settings:

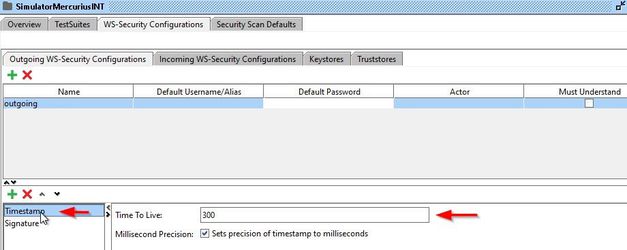


parts more in detail:

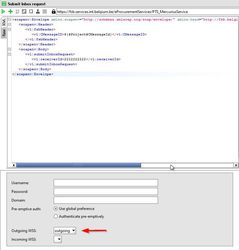
Body  
<http://schemas.xmlsoap.org/soap/envelope/>

Timestamp  
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>

timestamp settings (notice the order: timestamp first, signature next):



authentication settings on request:



## Connecting to the Mercurius TEST environment - FAQ

### How send testfiles to Mercurius (XML – PEPPOL)?

[how send testfiles to Mercurius.pdf](how%20send%20testfiles%20to%20Mercurius.pdf)

### How can I verify my testfile has arrived on Mercurius TEST?

[how can i verify my testfile has arrived on Mercurius TEST without errors.pdf](how%20can%20i%20verify%20my%20testfile%20has%20arrived%20on%20Mercurius%20TEST%20without%20errors.pdf)

### How can I validate my XML file for PEPPOL compliancy?

[How can i validate my XML file for PEPPOL compliancy.pdf](How%20can%20i%20validate%20my%20XML%20file%20for%20PEPPOL%20compliancy.pdf)

### How get access to Mercurius ?

[How get access to Mercurius.pdf](How%20get%20access%20to%20Mercurius.pdf)

**Identification of suppliers and customers in Mercurius**

[Identification of suppliers and customers in Mercurius](https://dwsoftbvba.sharepoint.com/Gedeelde%20%20documenten/peppol/simpl.ePrior/Simpl.ePrior%20bundle/Identification%20of%20suppliers%20and%20customers%20in%20Mercurius.pdf)

**Where can i verify receiving capabilities of a Belgian customer**

[Where can i verify receiving capabilities of a Belgian customer](https://dwsoftbvba.sharepoint.com/Gedeelde%20%20documenten/peppol/simpl.ePrior/Simpl.ePrior%20bundle/Where%20can%20i%20verify%20receiving%20capabilities%20of%20a%20Belgian%20customer.pdf)

**Acceptance testing when messages arrive on Mercurius**

[Acceptance testing when messages arrive on Mercurius](https://dwsoftbvba.sharepoint.com/Gedeelde%20%20documenten/peppol/simpl.ePrior/Simpl.ePrior%20bundle/Acceptancetesting%20when%20messages%20arrive%20on%20Mercurius.pdf)

# Connecting to Mercurius PROD

## Endpoint

### Mercurius PROD environment (FSB service at BOSA DT)

see: [S331-MercuriusService (Simpl.ePRIOR)](http://dtservices.bosa.be/nl/services/fsb/web-services-families/eprocurementservices-s294/documentatie-eprocurementservices-s294#S331-MercuriusService%20(Simpl.ePRIOR))

notice: previous URL is subject to change when connecting through one or more integrators. In this case the URL will be provided by the integrator.

remarks:

same remarks apply as specified in TEST when connecting directly to FSB.

current status:

* PEPPOL IN connection available
* PEPPOL OUT connection not available yet (planned for December 2018: to be confirmed)
* PEPPOL BIS V3 support not available yet (planned for March 2019: to be confirmed)
* PEPPOL BIS IR 63A document type (planned: to be confirmed)

# Security

To limit access only to documents concerning specific public entities, common name matching is performed between calling certificate (the certificate used by the integrator/public entity connecting to FSB) and common name linked to the incoming documents based on the enterprise number in the Mercurius configuration. This means public entities can only retrieve documents they have access for. If there is no match on the CN, an error is returned in the soap message.

example (of the error received in this case):



Remark:

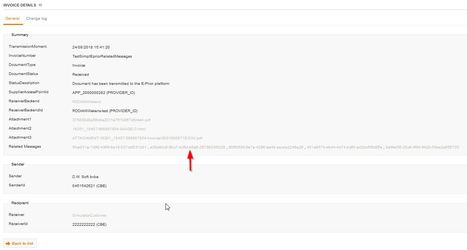
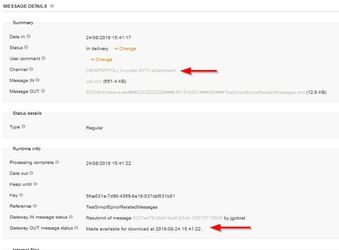
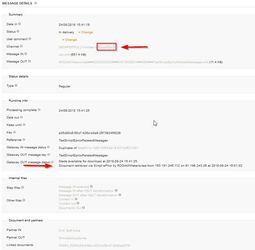
It seems sometimes we receive some other error message in this scenario:



# Limitations

* no testing on allowed documents types in sync. when sending SOAP. Valid files are checked afterwards and MLR will be returned in case of non-conformity. However, there is validity check on the file according to the filetype specified in the payload (e.g. CustomizationID in invoice 4A and 5A). To accomplish this, we use current schematrons available (e.g. PEPPOL BIS schematrons). Non-compliancy to this schematrons will return an MLR. Only xml syntax errors and UBL 2.1 invalidity will be reported back in sync as soap response "false".
* …

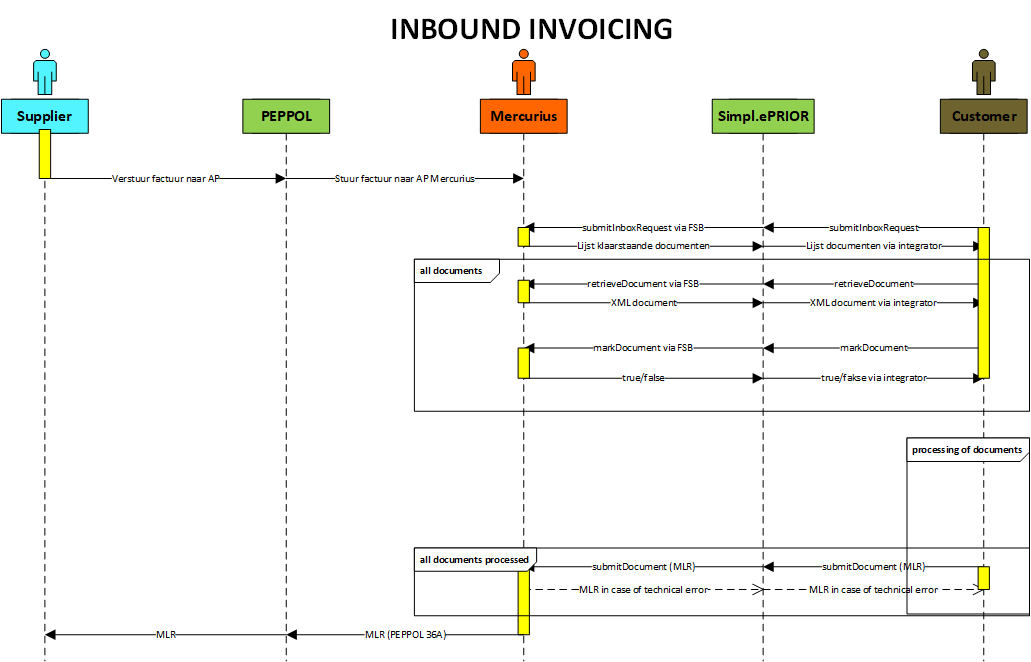
# Mercurius Portal changes

1. Change in document detail and message details:  
   It is possible to track the format of the message based on the channel name (as indicated below)
   1. The second message (in more than 95% of all documents) on the document detail form is the Simpl.ePRIOR message (every incoming document is presented in both ePrior and Simpl.ePRIOR format)  
      
   2. Details on the ePrior message (notice there is no indication about retrieving the document via the Simpl.ePRIOR interface):  
      
   3. Details on the Simpl.ePRIOR message (notice there is an indication about retrieving the document via the Simpl.ePRIOR interface):  
      

# Other scenarios

Previous descriptions apply to inbound invoicing. This section gives the necessary details on new scenarios added to the Simpl.ePRIOR interface.

## Inbound Invoicing



Remark:

In case of technical error when sending MLR to Mercurius (e.g. documentid not found), Mercurius is instrumented with logic to return an MLR to notify this situation. This is not done synchronously but is presented as document type 301 in the next SubmitInboxRequest command for retrieving and processing. In this case, no message is send to the supplier.

## Outbound Invoicing (target implementation date 31/12/2018)

### Concept

Outbound invoicing consists of sending electronic invoices and/or credit notes to other participants in a G2G and/or G2B scenario. This means technically sending PEPPOL BIS v2 4A Invoice and Billing 5A (invoice / credit note). Mercurius (Simpl.ePRIOR) will automatically add PEPPOL BIS v2 MLR 36A as receiving capability to a public entity when sending out the first outbound invoice. Doing so, makes it possible for receivers to respond to public entities when processing of the document is done (result might be accepted or rejected).

### Logic

* Document is send to Mercurius using SubmitDocument in the Simpl.ePRIOR interface
* Valid UBL/XML file verification is done synchronously in the soap connection
* Document type validation is done asynchronously based on CustomizationID and MLR containing error message is returned in case of non-compliancy
* Mercurius checks if receiver is defined as public entity receiver known by Mercurius: in this case, the message is rerouted internally as inbound invoice (G2G)
* When receiver is not known by Mercurius, the platform identifies receiving capabilities of the receiver on the PEPPOL network (SML/SMP) and determines APP\_ID (Access Point identification) of the receiver and sends message over the PEPPOL network. If no valid receiving capabilities are found in SML/SMP, Mercurius generates automatically an MLR send to the sender specifying no valid identifier was given. At some later stage, Hermes will be able to treat those outgoing messages for Belgian receivers if sufficient information is available in the message (eg. E-mail address).
* Notice that attachments will be treated based on the same logic “soap with attachment” as specified earlier. By doing so, the attachments added after the soap envelope will be inserted at the right place in the outgoing XML message and transmitted over the PEPPOL network.

### Formats allowed PEPPOL BIS v2:

#### Invoice:

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns010:ver2.0:extended:urn:www.peppol.eu:bis:peppol5a:ver2.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:www.cenbii.eu:profile:bii05:ver2.0</cbc:ProfileID>
* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns010:ver2.0:extended:urn:www.peppol.eu:bis:peppol4a:ver2.0</cbc:CustomizationID>  
   <cbc:ProfileID>urn:www.cenbii.eu:profile:bii04:ver2.0</cbc:ProfileID>

#### Credit Note:

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns014:ver2.0:extended:urn:www.peppol.eu:bis:peppol5a:ver2.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:www.cenbii.eu:profile:bii05:ver2.0</cbc:ProfileID>

#### MLR:

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns071:ver2.0:extended:urn:www.peppol.eu:bis:peppol36a:ver1.0</cbc:CustomizationID><cbc:ProfileID>urn:www.cenbii.eu:profile:bii36:ver2.0</cbc:ProfileID>

### Formats allowed PEPPOL BIS v3 (available 31/3/2019):

#### Invoice:

* <cbc:CustomizationID>urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:fdc:peppol.eu:2017:poacc:billing:01:1.0</cbc:ProfileID>

#### Credit Note:

* <cbc:CustomizationID>urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:fdc:peppol.eu:2017:poacc:billing:01:1.0</cbc:ProfileID>

#### MLR:

* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns71a:mlr:bis3-1.0</cbc:CustomizationID><cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis36a:mlr:bis3-1.0</cbc:ProfileID>

### Impact on portal

There will be minimal impact on the Mercurius portal. The same structure on documents and messages will be used. The only difference is that the sender can be a public entity and the receiver can be a private company (although G2G is also possible).

Status of outgoing documents.:

* received: means received by Mercurius and ready to be sent
* sent: when positive MDN is received, status is changed to "sent" (in case of G2G, since not sent over the PEPPOL network and if valid message, this applies immediately)
* rejected: 'RE' received as MLR response
* processed: 'AP' received as MLR response
* in the future when PEPPOL BIS v2 Invoice Response 63A is received, the ENECE name as status from UNECE code list 4343 (example: In Process, Under Query, …) based on ResponseCode in the IR will be used

### Information how SBDH is constructed

When the message will be send over the PEPPOL network, the content of the SBDH will be constructed based on:

* The SENDER information in the SBDH is taken from:
  + The 1st instance of /Invoice/AccountingSupplierParty/Party/PartyIdentification/ID if it exists
  + Else /Invoice/AccountingSupplierParty/Party/EndpointID
* The RECEIVER information in the SBDH is taken from:
  + The 1st instance of /Invoice/AccountingCustomerParty/Party/PartyIdentification/ID if it exists
  + Else /Invoice/AccountingCustomerParty/Party/EndpointID

### Final remarks

* To avoid an eternal exchange on status messages, Mercurius is equipped with some alert message stopping the Ping-Pong situation in this case. When a sender is sending some malformed status message (e.g. MLR) and the receiver is not able to process thus responding with some other status message where once again the original sender is not capable of understanding, Mercurius will stop this deadlock situating noticing Mercurius support of this situation. Mercurius support will investigate and take the necessary actions.
* When Mercurius needs to send status messages to public entities, we use the normal noticing order (1) send MLR (if capable of receiving MLR) (2) send mail with content equal to MLR to email address specified in sender in the outgoing invoice (if not present we might consider the technical email address for the sending entity as known by Mercurius) (3) status info can be consulted in the Mercurius portal

## Outbound e-Ordering (Target implementation date 31/12/2018)

### Concept

Outbound ordering consists of sending electronic orders to other participants in a G2B scenario. This means technically sending PEPPOL BIS v2 3A Order only and Ordering 28A. Mercurius (Simpl.ePRIOR) will automatically add PEPPOL BIS v2 28A as receiving capability to a public entity when sending out the first outbound order 28A. Doing so, makes it possible for receivers to respond to public entities when processing of the document is done (result might be accepted or rejected or accepted with remarks on orderlines).

### Logic

* Document is send to Mercurius using SubmitDocument in the Simpl.ePRIOR interface
* Valid UBL/XML file verification is done synchronously in the soap connection
* Document type validation is done asynchronously based on CustomizationID and MLR containing error message is returned in case of non-compliancy
* Mercurius identifies receiving capabilities of the receiver on the PEPPOL network (SML/SMP) and determines APP\_ID (Access Point identification) of the receiver and sends message over the PEPPOL network. If no valid receiving capabilities are found in SML/SMP, Mercurius generates automatically an MLR send to the sender specifying no valid identifier was given. At some later stage, Hermes will be able to treat those outgoing messages for Belgian receivers if sufficient information is available in the message (eg. E-mail address).
* Notice that attachments will be treated based on the same logic “soap with attachment” as specified earlier. By doing so, the attachments added after the soap envelope will be inserted at the right place in the outgoing XML message and transmitted over the PEPPOL network.
* Format PEPPOL BIS v2 3A is used when no response is expected, else use PEPPOL BIS v2 28A for sending. Notice that the format is the same on both formats except CustomizationID and ProfileID. Notice that OrderAgreement can be used to send a response when outgoing order 3A is used.

### Formats allowed PEPPOL BIS v2:

#### Order:

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns001:ver2.0:extended:urn:www.peppol.eu:bis:peppol28a:ver1.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:www.cenbii.eu:profile:bii28:ver2.0</cbc:ProfileID>
* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns001:ver2.0:extended:urn:www.peppol.eu:bis:peppol3a:ver2.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:www.cenbii.eu:profile:bii03:ver2.0</cbc:ProfileID>

#### OrderResponse (on format 28A sent):

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns076:ver2.0:extended:urn:www.peppol.eu:bis:peppol28a:ver1.0</cbc:CustomizationID><cbc:ProfileID>urn:www.cenbii.eu:profile:bii28:ver2.0</cbc:ProfileID>

#### OrderAgreement

* <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biitrns110:ver1.0:extended:urn:www.peppol.eu:bis:peppol42a:ver1.0</cbc:CustomizationID><cbc:ProfileID>urn:www.cenbii.eu:profile:bii42:ver1.0</cbc:ProfileID>

### Formats allowed PEPPOL BIS v3 (available 31/3/2019):

#### Order:

* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns01a:order:bis3-1.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis28a:ordering\_only:bis3-1.0</cbc:ProfileID>
* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns01a:order:bis3-1.0</cbc:CustomizationID>  
  <cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis03a:order\_only:bis3-1.0</cbc:ProfileID>

#### OrderResponse (on format 28A sent):

* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns76a:order\_response:bis3-1.0</cbc:CustomizationID><cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis28a:ordering:bis3-1.0cbc:ProfileID>

#### OrderAgreement

* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns110a:order\_agreement:bis3-1.0</cbc:CustomizationID><cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis42a:order\_agreement:bis3-1.0</cbc:ProfileID>

### Impact on portal

Order is a separate section in the Mercurius portal. The same structure on documents and messages will be used as on invoicing although some elements are specific for ordering.

Status of outgoing documents.:

* received: means received by Mercurius and ready to be sent
* sent: when positive MDN is received, status is changed to "sent"
* rejected: 27 received as OrderResponse (no orderliness sent)
* processed: 29 received as OrderResponse (no orderliness sent)
* processed with remarks: 30 received as OrderResponse (all orderliness must be sent with/without remarks)

### Information how SBDH is constructed

When the message will be send over the PEPPOL network, the content of the SBDH will be constructed based on:

* The SENDER information in the SBDH is taken from:
  + The instance of /Order/BuyerCustomerParty/Party/PartyIdentification/ID if it exists
  + Else /Order/BuyerCustomerParty/Party/EndpointID
* The RECEIVER information in the SBDH is taken from:
  + The 1st instance of /Order/SellerSupplierParty/Party/PartyIdentification/ID if it exists
  + Else /Order/SellerSupplierParty/Party/EndpointID

### Final remarks

* When Mercurius needs to send status messages to public entities, we use the normal noticing order (1) send MLR (if capable of receiving MLR) (2) send mail with content equal to MLR to email address specified in sender in the outgoing invoice (if not present we might consider the technical email address for the sending entity as known by Mercurius) (3) status info can be consulted in the Mercurius portal

## PEPPOL document type 63A Invoice Response (Target implementation date 31/12/2018)

### Concept

Outbound Invoice Response consists of sending electronic responses to other sending participants in a G2B scenario. This means technically sending PEPPOL BIS v2 63A Invoice Response. Before Mercurius (Simpl.ePRIOR) is capable of sending this document type, a participant sending invoices to public entities will have to add PEPPOL BIS v2 63A as receiving capability. Doing so, makes it possible for public entities to respond to senders when processing of the document (invoice/credit note) has started. This document format is capable of sending multiple status responses depending on the processing level of the document by a public entity (and is more adequate over the MLR).

### Logic

* Document is send to Mercurius using SubmitDocument in the Simpl.ePRIOR interface
* Valid UBL/XML file verification is done synchronously in the soap connection
* Document type validation is done asynchronously based on CustomizationID and MLR containing error message is returned in case of non-compliancy. Notice: although we are trying to send out Invoice Response, Mercurius is still using MLR to report back technical issues to the sending public entity)
* Mercurius identifies receiving capabilities of the receiver on the PEPPOL network (SML/SMP) and determines APP\_ID (Access Point identification) of the receiver and sends message over the PEPPOL network. If no valid receiving capabilities are found in SML/SMP, Mercurius generates automatically an MLR send to the sender specifying no valid identifier was given. At some later stage, Hermes will be able to treat those outgoing messages for Belgian receivers if sufficient information is available in the message (eg. E-mail address).

### Formats allowed PEPPOL BIS v2:

#### InvoiceResponse:

* <cbc:CustomizationID>urn:www.peppol.eu:transaction:biitrns111:ver1.0</cbc:CustomizationID><cbc:ProfileID>urn:www.peppol.eu:profile:bis63a:ver1.0</cbc:ProfileID>

### Formats allowed PEPPOL BIS v3 (available 31/3/2019):

#### InvoiceResponse:

* <cbc:CustomizationID>urn:fdc:peppol.eu:poacc:trns111a:invoice\_response:bis3-1.0</cbc:CustomizationID><cbc:ProfileID>urn:fdc:peppol.eu:poacc:bis63a:invoice\_response:bis3-1.0</cbc:ProfileID>

### Impact on portal

* Details about InvoiceResponse can be consulted on the original received document by requesting message details in Related Messages (almost everytime last message in the list). Notice however the difference between MLR and IR. The difference can be seen (without consulting the detail of the message) in the channel references in the message details pane.

### Information how SBDH is constructed

When the message will be send over the PEPPOL network, the content of the SBDH will be constructed based on:

* The SENDER information in the SBDH is taken from:
  + The instance of /ApplicationResponse/SenderParty/PartyIdentification/ID if it exists
  + Else /ApplicationResponse/SenderParty/EndpointID
* The RECEIVER information in the SBDH is taken from:
  + The 1st instance of /ApplicationResponse/ReceiverParty/PartyIdentification/ID if it exists
  + Else /ApplicationResponse/ReceiverParty/EndpointID

### Final remarks

* When Mercurius needs to send technical status messages to public entities, we use the normal noticing order (1) send MLR (if capable of receiving MLR) (2) send mail with content equal to MLR to email address specified in sender in the outgoing invoice (if not present we might consider the technical email address for the sending entity as known by Mercurius) (3) status info can be consulted in the Mercurius portal

## PEPPOL BIS v3 document types (Target implementation date 31/3/2019)

## PEPPOL BIS v2 mapping to PEPPOL BIS v3 (Target implementation date 31/3/2019)

## THE FUTURE (EPROCUREMENT: the next step)

