



PONTON
WE ARE THE 2 IN B2B

PONTON X/P 3.6.0 Backend Integration Guide

Version: 9

Date: 16-Oct-2018

Copyright Notice



This document is the confidential and proprietary information of PONTON GmbH ("Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with PONTON GmbH.

Table of Contents

1.	About this document.....	4
2.	Integration Scenarios.....	5
2.1.	Hotfolder Adapter	5
2.2.	HTTP Adapter	5
2.3.	Webservice Adapter	6
2.4.	Database Adapter	6
2.5.	Adapter API.....	6
3.	Outbound XML documents.....	7
3.1.	Data extraction from Backend-Envelope.....	7
3.1.1.	Required Data	8
3.1.2.	Optional Data	8
3.2.	Data extraction from custom envelope	9
3.2.1.	Optional Data	9
3.3.	Data extraction from document.....	10
3.3.1.	Message Type identification	10
3.3.2.	Mapping Definition.....	11
3.3.3.	Packaging Parameters	12
3.4.	Data supplied by API.....	13
3.5.	Sender / Receiver Identification by Hotfolders	13
4.	Other outbound documents	15
4.1.	EDIFACT.....	15
4.2.	X12.....	15
4.3.	Binary Data	16
5.	Sending attachments	17
6.	Inbound documents	18

1. About this document

The purpose of this document is to describe possible integration scenarios and integration adapters that are available from Ponton.

Processing of XML based messages is explained as well as the special treatment of EDIFACT and X12 messages.

2. Integration Scenarios

2.1. Hotfolder Adapter

The Hotfolder Adapter provides a quick and easy integration option that should work with almost any backend solution. The basic principle is a document exchange based on a shared file system.

Any inbound document is stored in an “inbox” folder, where the backend software can pick it up and delete it as needed. For this to work, a scheduled scanning of the “inbox” folder is needed.

Outbound documents have to be stored in an “outbox” folder. The Hotfolder Adapter scans this folder on a configurable interval. Any found document is transmitted and deleted automatically.

For better control over the outbound transfers there is an option to acknowledge the outbound transmissions with an acknowledgement document. This XML document (ERP-Acknowledgement) is stored in a special “acknowledgement-inbox” and can be parsed for transmission results.

To avoid processing of incomplete files, it is recommended to create outbound documents with a temporary file-extension (.tmp) and rename it after the creation.

For inbound documents the same practise is used by the Hotfolder Adapter, therefore the backend software should only process file with XML extension.

2.2. HTTP Adapter

The Http Adapter is a simple interface for transmitting documents via Http POST requests.

Inbound documents are transmitted to a Http port of the backend software.

Outbound documents need to be transmitted by the backend software as Http Post to the Http Adapter URL. The result is returned immediately as Http response.

2.3. Webservice Adapter

Webservices are a standard way of providing HTTP based services. Similar to a remote procedure call, a service is called with parameters (like a document) and a response is returned.

This is a very easy integration possibility when the backend solution is based on current Microsoft products.

2.4. Database Adapter

Most backend systems use some kind of database to store the business data. The Database Adapter provides a possibility to interface directly with the database tables of the backend system.

Incoming XML messages are mapped into database tables by applying mapping rules.

Outbound XML messages are created by the Database Adapter from database tables also by applying mapping rules.

A strong scripting language allows complicated adjustments and modifications between XML and database representations.

2.5. Adapter API

For tightest integration there is a Java based API that allows message exchange using a publish/subscribe approach. The API handles all the HTTP based communication between the Messenger and the Adapter, therefore it is a very simple API without high learning curve.

3. Outbound XML documents

When outbound XML documents are prepared in the backend solution there are some issues to be solved. Depending on the documents to be exchanged there might not be any data in the document that could be used for control of the transport.

Most important is the identification of Message Type, Sender and Receiver. Without this information it is not possible to transmit outbound documents.

There are four methods of supplying the needed information:

- using an xml backend-envelope
- using data from a custom envelope
- using data from the document itself
- using the API

3.1. Data extraction from Backend-Envelope

The basic idea of this approach is to wrap additional XML elements around the actual business document. These additional elements are referred to as the Backend-Envelope.

The Messenger will fetch transport relevant data from this Backend-Envelope and finally extract the business document. The Backend-Envelope only exists between the Adapter and the Messenger.

It will never be transmitted to an external partner.

Table 1 Structure of the Backend-Envelope

```
<?xml version="1.0"?>
<Envelope>
  <TransmissionInformation>
    <TransmissionCharacteristics TransmissionMode="BestEffort">
      <TransferID></TransferID>
      <ConversationID/>
    </TransmissionCharacteristics>
    <TransmissionOrganisationIdentifiers>
      <SenderOrganisation>internal id of local partner</SenderOrganisation>
      <ReceiverOrganisation>internal id of remote
partner</ReceiverOrganisation>
    </TransmissionOrganisationIdentifiers>
  </TransmissionInformation>
  <Payload>
    <MessageMetaData>
      <DocumentInfo MessageName="PurchaseOrder" TestFlag="Production">
        <LogInfo>Will be displayed in the AdminTool</LogInfo>
        <DTDVersionNumber>V1R00</DTDVersionNumber>
      </DocumentInfo>
    </MessageMetaData>
  </Payload>
</Envelope>
```

```
        </DocumentInfo>
    </MessageMetaData>
    <Message>
    \\
    <!-- Business part of message -->
    \\
        </Message>
    </Payload>
</Envelope>
```

3.1.1. Required Data

SenderOrganisation needs to match the “internal ID” of the sending partner as defined in the Messenger web interface.

ReceiverOrganisation needs to match the “internal ID” of the receiver as defined in the Messenger web interface.

MessageName and **DTDVersionNumber** need to match the attributes `MessageType` and `MessageVersion` found in the schema-set definition files of the Messenger distribution. (default location `/xmlpipe/webroot/WEB-INF/config/Schemata/`)

3.1.2. Optional Data

The **TransferID** element might contain a globally unique document id provided by the backend system for easier referencing of transfers. The Messenger will create a new unique ID when an empty element is provided.

The **ConversationID** element might contain an ID of a business process. If no ID is provided then the Messenger will create a new ID.

LogInfo can optionally contain any information that might be useful to either sender or receiver. The content of this element is shown in the Message-Monitor. The text is also transmitted to the receiver therefore the Message-Monitor of the receiver will display the same text.

TestFlag can be either “Production” or “Test”. This is also displayed in the Message-Monitor. Additionally it is possible to process Test Messages differently than Production Messages at the receiver side.

3.2. Data extraction from custom envelope

A custom envelope is supposed to belong to the actual payload; this means it is also transmitted to the receiver.

The envelope is identified by the following algorithm:

- Root-element name has to match
- If a namespace is defined, it has to match
- If a schemalocation is defined, the schemaLocation or noNamespaceSchemaLocation attribute has to end with this value
- All defined mapping rules have to reference existing elements or attributes

If multiple envelopes match all these conditions, then the first match is used. This case should be prevented for example with a unique schemalocation value.

Please note that the envelope is also verified during the xml validation step. So a wrongly identified envelope can result in validation errors.

If not all required information is found in the envelope, then the data extraction from the payload is performed in addition.

3.2.1. Optional Data

- Test
- SenderId
- ReceiverId
- MessageId
- MessageType
- MessageVersion
- SchemaSet
- LogInfo
- TestFlag

3.3. Data extraction from document

This requires a two-step process. First the message type needs to be identified, if it is not yet known due to an envelope. Based on the message type, a mapping definition is used to extract data from the XML.

3.3.1. Message Type identification

Identification is accomplished in the order as mentioned here, so if multiple possibilities would match, only the first match is relevant. Note that the identification is restricted to document types defined in the agreement belonging to the sender and receiver combination.

All values used for identification are defined in the schema set definition files found at `/xmlpipe/config/Schemata/*.xml`

The identification process depends on the use of namespaces in the payload:

Payload with Namespace

Root Element has match SchemaSet/Schema/RootElement or

SchemaSet/Schema/@MessageType

The **Namespace** attribute of the root element in the XML document **has to match** the value of the element

SchemaSet/Schema/Namespace

Identification by schemaLocation is attempted, if the message type is not precise at this point.

The location part of the schemaLocation attribute of the root element in the XML document is compared to different values to find a match. (only the location for the matching namespace is considered)

- Exact match to SchemaSet/Schema/@Name (case is ignored)
- Filename part of schemaLocation matches SchemaSet/Schema/SchemaFile

If the schemaLocation is not defined or there was no match, then the first match based on the Root Element is used.

Payload without Namespace

Root Element has match SchemaSet/Schema/RootElement or
SchemaSet/Schema/@MessageType

Identification by noNamespaceSchemaLocation is attempted, if the message type is not precise at this point.

- Exact match to SchemaSet/Schema/@Name (case is ignored)
- Filename part of schemaLocation matches SchemaSet/Schema/SchemaFile

Identification by DTDReference is attempted if message type is not precise at this point.

If there is a doctype reference like

```
<!DOCTYPE Envelope SYSTEM "PurchaseOrderV1R10.dtd">
```

in the XML document, the value will be compared to different values:

- Exact match to SchemaSet/Schema/DtdFile
- Exact match to SchemaSet/Schema/@Name

If the noNamespaceSchemaLocation or the DTDReference is not defined or there was no match, then the first match based on the Root Element is used.

3.3.2. Mapping Definition

If data should be extracted from the document it is required to define an EnvelopeMapping definition for each Message type in the Schemata configuration file.

Table 2 Example mapping

```
<Schema Name="Order.xsd" MessageType="Order" MessageVersion="1.00">  
  <Namespace></Namespace>  
  <DisplayName>Order 1.0</DisplayName>  
  <SchemaFile>order10.xsd</SchemaFile>  
  <XSLFile>order10.xsl</XSLFile>
```

```

<EnvelopeMapping>
  <LocalPartyIdXPath>/Order/SenderID</LocalPartyIdXPath>
  <RemotePartyIdXPath>/Order/ReceiverID</RemotePartyIdXPath>
  <MessageIdXPath>/Order/@RefID</MessageIdXPath>
</EnvelopeMapping>
</Schema>
    
```

Please note that the MessageId mapping is optional and should be left out if not used.

The mapping definition follows XPATH syntax, however only simple Element and Attribute content can be referenced

3.3.3. Packaging Parameters

If data should be overwritten in the packaging envelope, it is possible to define those values in the Schemata configuration file under PackagingParameters. So far the following elements can be overwritten:

Element name in Schema definition	XPath in AS4 Envelope relative to /Envelope/Header/Messaging/UserMessage	Comment
RoleFrom	/PartyInfo/From/Role	
RoleTo	/PartyInfo/To/Role	
Service	/CollaborationInfo/Service	
ServiceType	/CollaborationInfo/Service/@type	
Action	/CollaborationInfo/Action	
ConversationId	/CollaborationInfo/ConversationId	
AgreementRef	/CollaborationInfo/AgreementRef	
AgreementRefType	/CollaborationInfo/AgreementRef/@type	
MessageProperties	/MessageProperties/Property /MessageProperties/Property/@name	List of MessageProperty elements containing a propertyName and value
ContentProperties	/PayloadInfo/PartInfo/PartProperties/Property /PayloadInfo/PartInfo/PartProperties/Property/@name	List of ContentProperty elements containing a propertyName and a value

Table 3 Example Packaging Parameters

```

<Schema
Name="urn:easeegas.eu:edigas:capacitytrading:offeredcapacitydocument:5:1:System
    
```

```

OperatorOfferedCapacity" MessageType="AMV" MessageVersion="5.1">
<Namespace>urn:easeegas.eu:edigas:capacitytrading:offeredcapacitydocument:5:1</
Namespace>
  <DisplayName>System Operator Offered Capacity</DisplayName>
  <SchemaFile>urn-easee-gas-eu-edigas-capacitytrading-
offeredcapacitydocument-5-1.xsd</SchemaFile>
  <DtdFile></DtdFile>
  <XSLFile></XSLFile>
  <RootElement>OfferedCapacity_Document</RootElement>
  <PackagingParameters>
    <Service>A04</Service>
    <Action>http://docs.oasis-open.org/ebxml-
msg/as4/200902/action</Action>
    <RoleFrom>ZSO</RoleFrom>
    <RoleTo>ZUJ</RoleTo>
    <ConversationId></ConversationId>
    <ContentProperties>
      <ContentProperty
propertyName="EDIGASDocumentType">AMV</ContentProperty>
    </ContentProperties>
    <MessageProperties>
      <MessageProperty propertyName="partDetails">some
value</MessageProperty>
    </MessageProperties>
  </PackagingParameters>
</Schema>

```

3.4. Data supplied by API

On an instance of a BackendMessage object there are several setter methods that allow defining everything that would otherwise be read from the Backend-Envelope. Therefore the same rules for the values apply as mentioned in chapter ['Data Extraction From Backend-Envelope'](#)

For details on the API usage have a look at the "AdapterProgrammingGuide.pdf".

3.5. Sender / Receiver Identification by Hotfolders

A special identification option exists when using the hotfolder adapter.

When the option "use partner subfolders" is activated, there will be a subfolder for each remote partner. Placing a document in that folder automatically defines the receiver of the document.

The sender is also defined in the configuration of that hotfolder.

With this option it is not necessary to have any Envelope-mapping at all.

4. Other outbound documents

4.1. EDIFACT

EDIFACT documents can be used with the Hotfolder-Adapter without any Backend-Envelope or Envelope-Mapping. The Message-Type and Version is automatically detected and also Sender and Receiver IDs are fetched from the EDIFACT document.

EDIFACT content that is extracted:

- S002.0004 – interpreted as SenderID
- S003.0010 – interpreted as ReceiverID
- S004.0020 – used as MessageID
- S009.0065 – interpreted as MessageType
- S009.0052 and S009.0054 – these values are concatenated to form the MessageVersion. The Schemaset name is constructed with a static “EDIFACT_” prefix and the MessageVersion appended.

4.2. X12

X12 documents can be used with the Hotfolder-Adapter without any Backend-Envelope or Envelope-Mapping. The Message-Type and Version is automatically detected and also Sender and Receiver IDs are fetched from the X12 document.

X12 content that is extracted:

- ISA06 – interpreted as the SenderID
- ISA08 – interpreted as the ReceiverID
- ISA13 – used as main part of the MessageID. As The ISA13 Interchange Control number is unique only for the sender, it is extended by the SenderID to build the MessageID.
- ISA12 – interpreted as Schemaset (without the zeros)

- ST01 — interpreted as the schema (=MessageType). The Transaction Set Header (ST) is a repetition element, but the messenger only considers the ST01 element in the first Transaction Set.

4.3. Binary Data

Binary data could be anything, therefore it is impossible to have any automatic handling of those documents. A Hotfolder with enabled partner-subfolders can be used to send out binary data.

The alternative is a specialized adapter that is able to fetch information from the binary content.

5. Sending attachments

It is possible to transmit additional files of any kind together with a business document. These attachments are not processed in any way.

The procedure of attaching files to a business document is the same for Hotfolder-Adapter and API usage:

A folder has to be created with the same name as the business document excluding the extension. Any file that is in this folder will be attached to the business document.

6. Inbound documents

The Messenger will deliver any inbound documents to the Adapter defined in the partner agreement. If the destination Adapter is not running for any reason, then the Messenger will queue the documents and deliver them as soon as the Adapter registers again with the Messenger.

The details of inbound integration depend very much on the used Adapter, so it is recommended to consult the Adapter documentation.

PONTON GmbH
Dorotheenstraße 64
GERMANY 22301 Hamburg
Web: <http://www.ponton.de>