

# Specification

## **EFET Box+ back end interface – Broker Box**

An interface to couple the Broker Back End System with an EFET Trader Box+.

Version 3.2

Date 2006-03-24

Written by

Ponton Consulting GmbH

Dorotheenstr. 60

22301 Hamburg

Contact:

Dr. Michael Merz

Tel.: +49 40 69 213 341

Mob.: +49 40 170 85 22 894

Email: [merz@ponton-consulting.de](mailto:merz@ponton-consulting.de)

# Content

|   |           |
|---|-----------|
| <b>Content .....</b>  | <b>2</b>  |
| <b>1 Scope of this document .....</b>                                 | <b>3</b>  |
| <b>2 Integration scenarios with Broker Back End Systems.....</b>      | <b>4</b>  |
| 2.1 Changes from EFETBox version 3.1 to 3.2 .....                     | 4         |
| 2.2 Architectural overview .....                                      | 4         |
| 2.3 Logical integration.....  | 4         |
| <b>3 Hotfolder Adapter based integration.....</b>                     | <b>6</b>  |
| 3.1 Sequence of actions .....   | 6         |
| 3.2 Viewpoint of the Broker Back End System.....                      | 6         |
| 3.3 Technical Requirements .....                                      | 6         |
| <b>4 Database Adapter (Ponton X/D) based integration .....</b>        | <b>8</b>  |
| <b>5 Web Service based integration .....</b>                          | <b>9</b>  |
| 5.1 About Web Services .....  | 9         |
| 5.2 Overview of the PONTON WSA (Web Service Adapter) .....            | 9         |
| 5.3 Installation and Configuration .....                              | 10        |
| 5.4 Usage .....   | 11        |
| 5.5 The WSDL (Web Service Description Language) File .....            | 11        |
| <b>6 Linking Buyer and Seller Trade Confirmations of a Deal .....</b> | <b>14</b> |
| <b>7 BoxResult specification.....</b>                                 | <b>15</b> |
| 7.1 Expected responses .....  | 15        |
| 7.2 Elements of the BoxResult .....                                   | 16        |
| 7.3 XML Schema for BoxResult .....                                    | 17        |
| <b>8 Valid Values and Enumerations .....</b>                          | <b>20</b> |

## 1 Scope of this document

This document addresses the system administrator in an energy trading company. The document is not written for the end users in the energy trading business.

This document describes ways to connect your Back End System with the EFET Broker Box+ Version 3.1 and 3.2.

The second chapter outlines the different integration options, which are detailed in the subsequent chapters. The sixth chapter gives information how to link to BCN documents together, so that they are displayed as one deal in the EFETBox+ user interface. Chapter 7 describes the Box Results produced by the EFETBox+. Box Results inform the Back End System about the status of the processed documents. In the final chapter, you can find a useful list of valid values and enumerations for creating your input data for the EFETBox+.

For further information on configuration and installation of the EFET Box+, please refer to the document 'EFETBox3.2\_InstallConfigGuide'.

## 2 Integration scenarios with Broker Back End Systems

The EFETBox provides the matching process for matching Trade Confirmations between counterparties and brokers.

The EFETbox is available as trader or broker box. The trader box matches Trade Confirmations both with the counterparty and an involved broker. Additionally, Broker Fee Information is matched between the trader and broker. The broker box sends Trade Confirmations and Broker Fee Information to the trader's EFETBox, which matches the documents and sends corresponding match results back to the broker.

This document describes the broker box. The trader box is described in the document 'EFETBox 3.2 BoxBackEndInterface TraderBox'.

There has to be a connection to the Broker Back End System so that the EFETBox can pick up Trade Confirmations and Broker Fee Information Documents and report back the status of the matching process (BoxResults).

This specification proposes a simple mechanism for integrating the EFETBox+ with a broker's back end system.

### 2.1 Changes from EFETBox version 3.1 to 3.2

With respect to the , , When upgrading to the EFETBox version 3.2 the following changes need to be considered:

- outbound interface<sup>1</sup>: The Broker Confirmation (BCN) and Cancellation (CAN) documents defined for the 3.1 EFETBox remain valid, except for one change in the BCN document: the section 'Account And Charge Information' has been removed.. Additionally, BCN documents for emissions deals will be accepted by the EFETBox. The schema reference in the XML files should be changed to 3.2 (i.e., the attribute "SchemaRelease" should have the value "2" instead of "1").
- inbound interface: The BoxResults, providing ETRM systems with status change of the received documents, remain unchanged to the EFETBox version 3.1.

### 2.2 Architectural overview

There are three parts to be considered for this integration:

- The Broker Back End System is the principal source of business data.
- The EFETBox+ performs the Electronic confirmation matching and it incorporates the associated protocol.
- The Ponton X/P Messenger handles the ebXML transfer details.

### 2.3 Logical integration

The Broker Back End System (on behalf of the user) initiates the message transfer to the trader's EFETBox.

The Broker Back End System pushes either a Broker Confirmation (BCN) or a Cancellation (CAN) document to the EFETBox.

The EFETBox then transfers the BCN or CAN document to the corresponding business party and executes the workflow process especially the confirmation matching (eCM).

The matching of the trader's CNF and BFI documents with the broker's BCN document is performed by the trader's EFETBox, the broker's EFETBox does not perform matching.

After the matching process has reached an end state, the EFETBox notifies the Broker Back End System about the result. Additionally, important intermediate steps, e.g., when a BCN has reached the status pending, are forwarded to the Broker Back End Systems.

---

<sup>1</sup> i.e., the sending of documents to the EFETBox

The interface between the EFETBox and the Broker Back End System, as outlined in figure 1 'EFETBox+ system overview', can be implemented

- By means of a Hotfolder Adapter, which uses the file system to swap messages between the backend system and the EFETBox, or
- Using the Database Adapter (Ponton X/D) for direct database access.
- Additionally, it is possible to develop an own adapter implementation. The document 'EFETBox 3.2 AdapterProgrammingGuide.pdf' gives detailed information and can be provided by Ponton Consulting on request.
- A procedural integration via the WebService Adapter.

In either case, it may be necessary to develop some form of extension to the Broker Back End System to ensure correct handling of the business processes.

The different options could be combined, e.g., using the Hotfolder Adapter for the communication to the EFETBox+ and using the Database adapter for the box results.

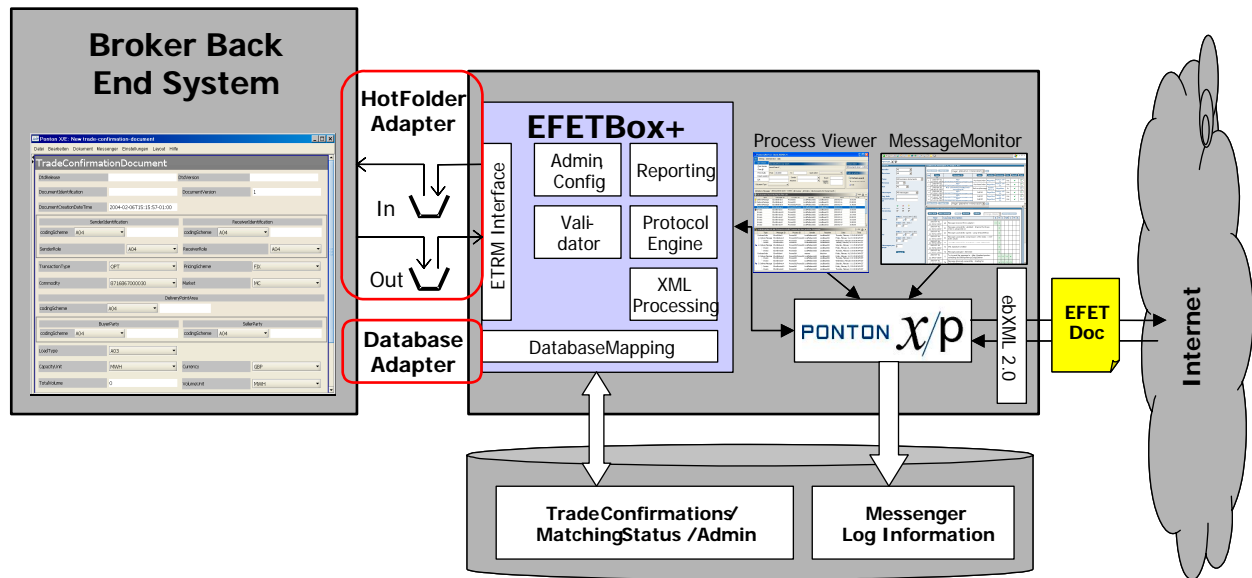


Figure 1: EFETBox+ system overview

### 3 Hotfolder Adapter based integration

The proposed integration solution is based on the exchange of data files via the operating system's file system.

#### 3.1 Sequence of actions

The Broker Back End System stores relevant documents (BCN or CAN) in an **outbox** folder. This can be triggered by a user request, or the Broker Back End System can export the documents on a periodic schedule. The xml schema for these documents are defined in the EFET 3.2 Standard (please refer to the EFET homepage: [www.efet.org](http://www.efet.org)).

The EFETBox scans this outbox based on a fixed schedule and a given filename pattern (\*.xml). The documents are removed from the outbox, validated and inserted into the workflow process.

In case of an error or a problem with the document's format, that prevents it from entering the workflow process, the document will be rejected and the file will be stored in a **failed** folder.

When the document reaches either an end states, a matching end state or other specific states the EFETBox generates a BoxResult document (see section 7 BoxResult specification' of this document a more information) and stores it in an **inbox** folder. The xml document for the BoxResults is defined in the section 7.3 'XML Schema for BoxResult'.

#### 3.2 Viewpoint of the Broker Back End System

After a document is exported from the Broker Back End System, the document is in one of the following four stages:

1. Not yet processed – the document is still present in the outbox folder
2. Rejected due to a syntax error – the document is in the failed folder
3. Pending – a BoxResult document is in the inbox folder
4. Finished with matching – a BoxResult document is in the inbox folder

The successful result of a *broker match* references the BCN document of the Broker Back End System and the trader's CNF document.

In case of a failure, the matching process yields a code and a description for the error in the BoxResult document.

When the EFETBox is confronted with mass processing, a throughput number should be defined. For incoming documents above this rate, an increased waiting time (in stage 1) has to be accepted.

#### 3.3 Technical Requirements

Since a typical file-system has no transactional features, there has to be a clear distinction when a file is created and deleted.

To prevent premature reading, the document initially has to be written with a temporary file extension (not ending in .xml). After it has been fully written to the file system, the document is renamed to its actual name with the file name extension ".xml". When testing under Windows 2000, import errors were noted with respect to premature reading.

After a file has been written to the input folders (either failed or inbox), the Broker Back End System has to process it. There is no notification (to the EFETBox) that the Result document has been read, so the EFETBox must not delete this information even though it created the file. Instead the Broker Back End System is free to delete these files after it has extracted the information.

The file names have to be unique, so that different documents can be stored in the same folder without one overwriting the other. A natural choice for the file name would be the unique ID of the document, given by its documentId plus its version number. We propose names of the form

```
BCN_20040917000001_Ponton-001.xml  
<DocumentID>-<Version>.xml
```

In order to prevent problems with the file system, special/reserved characters like "@", "/", "\", "&", "|", ">", "<", "%", "\$", ":", ";", "'", "\*", "?" should be replaced with an underscore character "\_". Please note also that localized characters like "ä", "ø", "ç" can also pose problems to some file systems.

## 4 Database Adapter (Ponton X/D) based integration

The Database Adapter – also available as independent product Ponton X/D – is a **connectivity tool** that integrates databases with XML middleware (here the EFETBox).

Integrating Ponton X/D with the EFETBox involves editing two to four parameters in a configuration file. After setting up an appropriate “mapping definition” file, basically any kind of XML message can be mapped to any number of databases/tables and vice versa. In a common scenario a set of staging tables is used as an interface between the ERP and the EFETBox.

Database access is set up easily with standard JDBC/ODBC interfaces. The database systems supported include IBM DB2, Oracle 7/8i/9i/10g, MS SQL Server, Informix, MySQL etc. Additional X/D features include support for complex value transformations, conditional mapping, fallback mapping, DB↔DB mapping, insert vs. update mode, flat file generation and more.

X/D can be customized with mappings for any XML schemas and arbitrary database products and schemas; currently default mappings are available for

- Broker Confirmations
- Cancellations
- BoxResults

The X/D DatabaseAdapter comes as a self-extracting executable with a general and a mapping manual.

To achieve basic connectivity you will only have to take two steps:

- Configure the EFETBox as a “Messenger” in X/D’s dbadapter.xml, e.g. for a default installation on the same host:

```
<Messenger>
  <AdapterID>XD_EFET</AdapterID>
  <Address>localhost</Address>
  <Port>8880</Port>
  <Path>/efet/AdapterService</Path>
</Messenger>
```

- From the EFET Box+ side you have to declare X/D as the default adapter used for inbound communication (EFETBox→backend database). This is done by changing one value in C:\Program Files\Ponton EFET Box\efet\webroot\WEB-INF\config\box\box.xml with a text editor: In the section

```
<BackOffice>
  <UsedAdapter Name="XD_EFET" />
</BackOffice>
```

enter the unique AdapterID you specified in the X/D’s dbadapter.xml (see above) and restart the EFET Box+.

Note: Ponton X/D’s numerous applications and configuration options are beyond the scope of this manual, refer to the X/D documentation for details.

For further information please refer to the document

[http://www.ponton-consulting.de/downloads/xd/PontonXD-LeifheitCase-2004-07\\_en.pdf](http://www.ponton-consulting.de/downloads/xd/PontonXD-LeifheitCase-2004-07_en.pdf)

or contact Ponton Consulting.

## 5 Web Service based integration

### 5.1 About Web Services

A Web service is a collection of protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards. OASIS and the W3C are the steering committees responsible for the architecture and standardization of web services. To improve interoperability between web service implementations, the WS-I organisation has been developing a series of profiles to further define the standards involved.

### 5.2 Overview of the PONTON WSA (Web Service Adapter)

The Web Service Adapter is yet another adapter for the EFET Box+. Since Web Services are request/response based, the WSA is a state full implementation which needs persistent message queues. The main components are the x/p adapter which connects to the EFET Box+ and the web service servlet, which provides the service. All messages are queued in the file system with references to them in the database.

Please note that PONTON only provides the server part of the Web Service. In order to use the Web Services, a client part needs to be developed within your company using your preferred web service technology. Ponton Consulting can support you in the development of such a customized web service client. Please note, that this support is not part of the helpdesk support services contracted through the Support and Maintenance Agreement between Ponton Consulting and EFETnet/EFETnet clients.

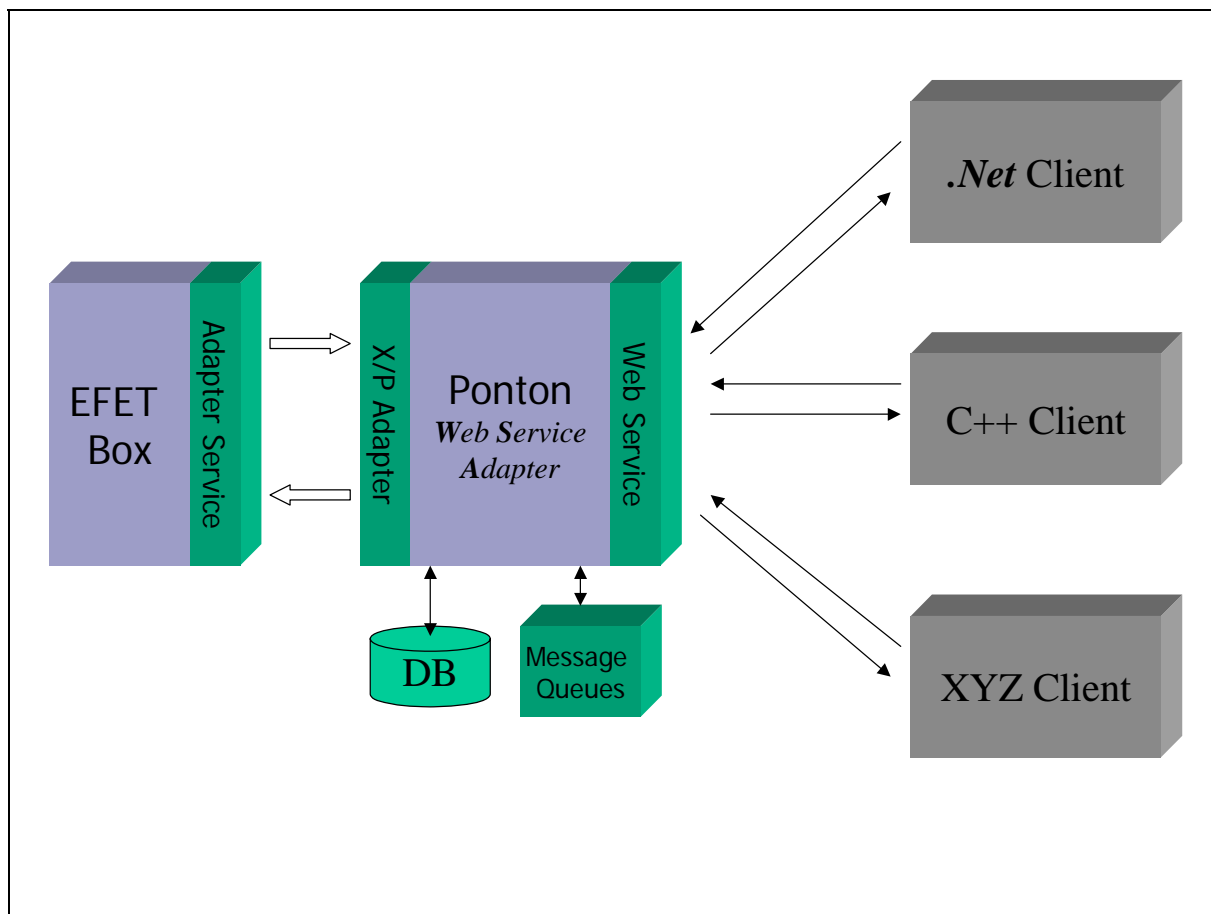


Figure 2: Overview of the Web Service Adapter components

### 5.3 Installation and Configuration

The Web-Application Archive (war-file) is not part of the regular EFETBox+ software distribution. You can download this Web-Application Archive from the site [www.ponton-consulting.de/efetbox/](http://www.ponton-consulting.de/efetbox/) in the section 'EFETBox 3.1/3.2 Additional Information'.

The following steps need to be performed for the installation of the Web Service Adapter for the EFET Box+:

- Stop the EFET Box+
- Copy the file "wsa.war" to the location "<install\_path\_efet\_box>\tomcat-5.0.28\webapps"
- Copy the JDBC driver file used for the EFET Box+ distribution to the location "<install\_path\_efet\_box>\tomcat-5.0.28\webapps\wsa\WEB-INF\lib"
- Start EFET Box+. (The server will unzip the "wsa.war" into a new folder "wsa")
- Edit the file "<install\_path\_efet\_box>\tomcat-5.0.28\webapps\wsa\WEB-INF\config\wsa\WSAConfiguration.xml" and fill out the elements marked in bold. Please make sure to replace the <EncryptedPassword> tag with a <Password> tag. It will be replaced by an encrypted one after restarting the EFET Box+. The information to be inserted is defined in the file "<install\_path\_efet\_box>\efet\webroot\WEB-INF\config\box\box.xml".

```

...
<Database>
  <Driver>oracle.jdbc.OracleDriver</Driver>
  <URL>jdbc:oracle:thin:@danbala:1521:danbala</URL>
  <User>efet31_04</User>
  <Password>clear_text_password</Password>
  <HibernateDialect>
net.sf.hibernate.dialect.Oracle9Dialect
</HibernateDialect>
...
</Database>
<Messenger>
  <MessengerHost>hostname_of_efet_box_machine</MessengerHost>
  <MessengerPort>port_of_efet_box_machine (Default is 8880)</MessengerPort>
...
  </Messenger>

```

- Edit the file "<install\_path\_efet\_box>\efet\webroot\WEB-INF\config\box\box.xml" and change the used Adapter to the value "WebServiceAdapter":

```

<BackOffice>
  <UsedAdapter Name="WebServiceAdapter"/>
</BackOffice>

```

- Restart the EFET Box+
- Open a browser on [http://<host\\_name\\_of\\_efet\\_box\\_machine>:8880/wsa/services/WebServiceAdapterPort?wsdl](http://<host_name_of_efet_box_machine>:8880/wsa/services/WebServiceAdapterPort?wsdl)

If the installation was successfully the WSDL (Web Service Description Language) XML-file should be displayed in the browser.

## 5.4 Usage

To develop a client for the WSA the WSDL file

["http://<host\\_name\\_of\\_efet\\_box\\_machine>:8880/wsa/services/WebServiceAdapterPort?wsdl"](http://<host_name_of_efet_box_machine>:8880/wsa/services/WebServiceAdapterPort?wsdl)

should be used. The Web Service provides three methods:

- *sendDocument*
- *getNextDocument*
- *getStatusForMessage*

Right now only the *sendDocument* and *getNextDocument* methods are implemented since a status request for a message is at the time not yet available. The most actual integrated development environment (IDE) provide wizards for the creation of Web Service Client applications.

## 5.5 The WSDL (Web Service Description Language) File

```
<?xml version="1.0" encoding="UTF-8" ?>
<wscdl:definitions targetNamespace="urn:WebServiceAdapter"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="urn:WebServiceAdapter" xmlns:intf="urn:WebServiceAdapter"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wscdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wscdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <wscdl:types>
    <schema targetNamespace="urn:WebServiceAdapter"
      xmlns="http://www.w3.org/2001/XMLSchema" >
      <complexType name="DocumentStatusResponse" >
        <sequence>
          <element name="messageId" nillable="true" type="xsd:string" />
          <element name="status" nillable="true" type="xsd:string" />
          <element name="statusCode" nillable="true"
            type="xsd:string" />
          <element name="statusDescription" nillable="true"
            type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="SendDocumentResponse" >
        <sequence>
          <element name="HttpCode" type="xsd:int" />
          <element name="xpCode" type="xsd:int" />
          <element name="Description" nillable="true" type="xsd:string"
            />
          <element name="MessageId" nillable="true" type="xsd:string" />
          <element name="ConversationId" nillable="true"
            type="xsd:string" />
          <element name="MessageTime" nillable="true" type="xsd:string"
            />
          <element name="MessageType" nillable="true" type="xsd:string"
            />
          <element name="SchemaSet" nillable="true" type="xsd:string" />
          <element name="SchemaVersion" nillable="true"
            type="xsd:string" />
          <element name="TransmissionProtocol" nillable="true"
            type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="XMLDocument" >
        <sequence>
          <element name="payload" type="xsd:base64Binary" />
        </sequence>
      </complexType>
      <complexType name="SendDocumentRequest" >
        <sequence>
          <element name="MessageId" nillable="true" type="xsd:string" />
          <element name="ConversationId" nillable="true"
            type="xsd:string" />
          <element name="SenderId" nillable="true" type="xsd:string" />
          <element name="ReceiverId" nillable="true" type="xsd:string"
            />
          <element name="MessageType" nillable="true" type="xsd:string"
            />
        </sequence>
      </complexType>
    </schema>
  </wscdl:types>
</wscdl:definitions>
```

```

        <element name="SchemaSet" nillable="true" type="xsd:string"/>
        <element name="SchemaVersion" nillable="true"
            type="xsd:string" />
        <element name="payload" type="xsd:base64Binary" />
    </sequence>
</complexType>
<element name="request" type="impl:SendDocumentRequest" />
<element name="sendDocumentReturn" type="impl:SendDocumentResponse" />
<element name="getNextDocumentReturn" type="impl:XMLDocument" />
<element name="messageId" type="xsd:string" />
<element name="getStatusForMessageReturn"
    type="impl:DocumentStatusResponse" />
</schema>
</wsdl:types>
- <wsdl:message name="getStatusForMessageResponse">
    <wsdl:part element="impl:getStatusForMessageReturn"
        name="getStatusForMessageReturn" />
</wsdl:message>
<wsdl:message name="getNextDocumentRequest" />
- <wsdl:message name="sendDocumentRequest">
    <wsdl:part element="impl:request" name="request" />
</wsdl:message>
- <wsdl:message name="getNextDocumentResponse">
    <wsdl:part element="impl:getNextDocumentReturn" name="getNextDocumentReturn"
        />
</wsdl:message>
- <wsdl:message name="getStatusForMessageRequest">
    <wsdl:part element="impl:messageId" name="messageId" />
</wsdl:message>
- <wsdl:message name="sendDocumentResponse">
    <wsdl:part element="impl:sendDocumentReturn" name="sendDocumentReturn" />
</wsdl:message>
- <wsdl:portType name="WebServiceAdapter">
    - <wsdl:operation name="sendDocument" parameterOrder="request">
        <wsdl:input message="impl:sendDocumentRequest"
            name="sendDocumentRequest" />
        <wsdl:output message="impl:sendDocumentResponse"
            name="sendDocumentResponse" />
    </wsdl:operation>
    - <wsdl:operation name="getNextDocument">
        <wsdl:input message="impl:getNextDocumentRequest"
            name="getNextDocumentRequest" />
        <wsdl:output message="impl:getNextDocumentResponse"
            name="getNextDocumentResponse" />
    </wsdl:operation>
    - <wsdl:operation name="getStatusForMessage" parameterOrder="messageId">
        <wsdl:input message="impl:getStatusForMessageRequest"
            name="getStatusForMessageRequest" />
        <wsdl:output message="impl:getStatusForMessageResponse"
            name="getStatusForMessageResponse" />
    </wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="WebServiceAdapterPortSoapBinding"
    type="impl:WebServiceAdapter">
    <wsdlsoap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http" />
    - <wsdl:operation name="sendDocument">
        <wsdlsoap:operation soapAction="" />
        - <wsdl:input name="sendDocumentRequest">
            <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />
        </wsdl:input>
        - <wsdl:output name="sendDocumentResponse">
            <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />
        </wsdl:output>
    </wsdl:operation>
    - <wsdl:operation name="getNextDocument">
        <wsdlsoap:operation soapAction="" />
        - <wsdl:input name="getNextDocumentRequest">
            <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />
        </wsdl:input>
        - <wsdl:output name="getNextDocumentResponse">
            <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />
        </wsdl:output>
    </wsdl:operation>
    - <wsdl:operation name="getStatusForMessage">
        <wsdlsoap:operation soapAction="" />
        - <wsdl:input name="getStatusForMessageRequest">
            <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />

```

```
</wsdl:input>
- <wsdl:output name="getStatusForMessageResponse">
  <wsdlsoap:body namespace="urn:WebServiceAdapter" use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
- <wsdl:service name="WServiceAdapterService">
- <wsdl:port binding="impl:WebServiceAdapterPortSoapBinding"
  name="WebServiceAdapterPort">
  <wsdlsoap:address
    location="http://verp:8880/wsa/services/WebServiceAdapterPort" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

## 6 Linking Buyer and Seller Trade Confirmations of a Deal

Each leg of a trade with the broker as intermediary is represented by a separate BCN document, i.e., the Broker Back End System creates a Broker Confirmation document for the seller and a separate one for the buyer, as defined by the EFET eCM Standard version 3.1 and 3.2.

Linking the Broker Confirmation of the buyer and seller leg would allow the display of the buyer and seller Broker Confirmation and the corresponding status information about the matching success as one deal.

The EFETBox can represent the buyer and seller leg of the same deal as a unit, if the "Document ID" of both trade confirmations is created in accordance to an additional business rule.

A general rule for creating the unique Document ID is defined within the EFET Standard version 3.1/3.2 as follows:

1. Document type abbreviation (e.g. "BCN" for Broker Confirmation)
2. DateCode (8 characters, in yyyyymmdd format), TradeDate (extract from BCN)
3. Locally & daily unique TradeID (min. 10 characters) of the sender side
4. "@"
5. Sender identification, i.e. domain name or EIC Code of the sender.

The additional rule for linking two Broker Confirmations further refines the 3<sup>rd</sup> point:

Two Broker Confirmations are linked to a deal in the Broker Box, if the two TradeIDs have the form

<TradeID>\_B (buyer leg) and <TradeID>\_S (seller leg)

and

the rest of the document IDs is equal.

Example:

BCN\_20040610\_1234567890\_B@11XELECTRABEL--Z

BCN\_20040610\_1234567890\_S@11XELECTRABEL--Z

## 7 BoxResult specification

### 7.1 Expected responses

Brokers are included in the matching process between trades with the introduction of the EFET 3.1 standard. Brokers send BCN documents to the trader's EFETBox, which matches the documents and returns Broker Match Notification documents (BMN). For a more detailed description of the business processes and error codes, please refer to the EFET eCM Standard version 3.1 or 3.2 (please refer to the EFET homepage: [www.efet.org](http://www.efet.org)).

Remark: The EFETBox typically generates more than one BoxResult for a given document: E.g., in case of a BCN document, first the "PENDING" and afterwards the "PRELIMINARY MATCHED" and "MATCHED" result. In case of a timeout the sequence of BoxResults can be "PENDING", "TIMED OUT" and finally "AMENDED".

#### 7.1.1 Box results for BCN documents

When the Broker Back End System sends a BCN document, the EFETBox generates a BoxResult document that reports the confirmation to be in broker state "PENDING" after the initial processing reaches this state. This document signals that the processing of the BCN document has started.

If the BCN document could not be transferred to the trader's EFETbox, a BoxResult will be generated that reports the confirmation in the state "NOT SENT".

If the BCN document was not accepted, e.g., because a duplicate BCN was sent, a BoxResult will be generated and informing the Broker Back End System about the reason for rejecting the document.

Once the broker's EFET Box has been informed about a broker match in form of a BMN, a BoxResult will be generated that reports the confirmation to be in state "MATCHED" or "PRELIMINARY MATCHED", in case the counterparty match has not yet occurred. This Matched-Result will reference both confirmations, from the trader's Trade Confirmation as well as from the broker's Broker Confirmation.

If the business process fails (because a document is rejected), a BoxResult will be generated that reports the confirmation to be in broker state "FAILED".

In case that the matching does not succeed in time, a BoxResult will be generated that reports the confirmation to be in cp state "TIMED OUT".

#### 7.1.2 Box results for BCN amendments

The Broker Back End System may send a follow up BCN document as an amendment to a previously sent document. This amendment may be accepted or rejected.

If the amendment is accepted, it supersedes the previous version of the document. In this case a BoxResult will be generated that reports the previous document to be in state "AMENDED".

If the amendment is not accepted, a BoxResult will be generated that reports the amendment document (the newer one) is not accepted and gives further details to the reason why the amendment failed.

#### 7.1.3 Box results for CAN documents

Furthermore, the Broker Back End System could send a CAN document (cancellation). The box can cancel the BCN document, or the confirmation may be in a state that cannot be cancelled, in which case the cancellation will fail.

In the first case, a BoxResult will be generated that reports the confirmation to be in state "CANCELLED".

In case of a failure, a BoxResult will be generated with a reference to the cancellation document and an explanation, why the confirmation could not be cancelled.

## 7.2 Elements of the BoxResult

The BoxResult has three mandatory elements:

- DocumentType identifies the document type (BCN or CAN) of the referenced document.
- Second is the DocumentID, which identifies the document that originates from the ETRM System and that has been stored in the outbox.
- The last mandatory element is Timestamp, which gives the local date and time when the document reached the end state.

Each BoxResult provides a State to inform about the state of the corresponding document. The possible values for BCN documents are: "PENDING", "MATCHED", "PRELIMINARY MATCHED", "AMENDED", "CANCELLED", "TIMED OUT", "FAILED" or "NOT SENT").

If the BoxResult concerns a Broker Confirmation, the optional element DocumentVersion is used together with the document ID to uniquely identify the document.

When the document has been sent to the trader's EFETBox, the ebXML message ID is included in the element ebXMLMessageId.

In case of a failure (the states "TIMED OUT", "FAILED" or "NOT SENT"), the optional element Reason provides detailed information about the origin of the failure. This element reuses the definition of the EFET standard's CoreElements.

Whenever a match – either preliminary or final – is notified to the back end system, the section Counterparty informs about the corresponding CNF document from the trader.

The following elements of the BoxResult xml schema are not used in the BoxResults of the broker box, they are only used by the trader box:

- BrokerState
- Broker
- MatchInformation
- BrokerMatchInformation

### 7.3 XML Schema for BoxResult

The XML schema for the BoxResult is also available at the site: <http://www.ponton-consulting.de/efetbox/>, section 'EFETBox 3.1/3.2 Additional Information', item 'EFETBox 3.1/3.2 Box Results Schema'.

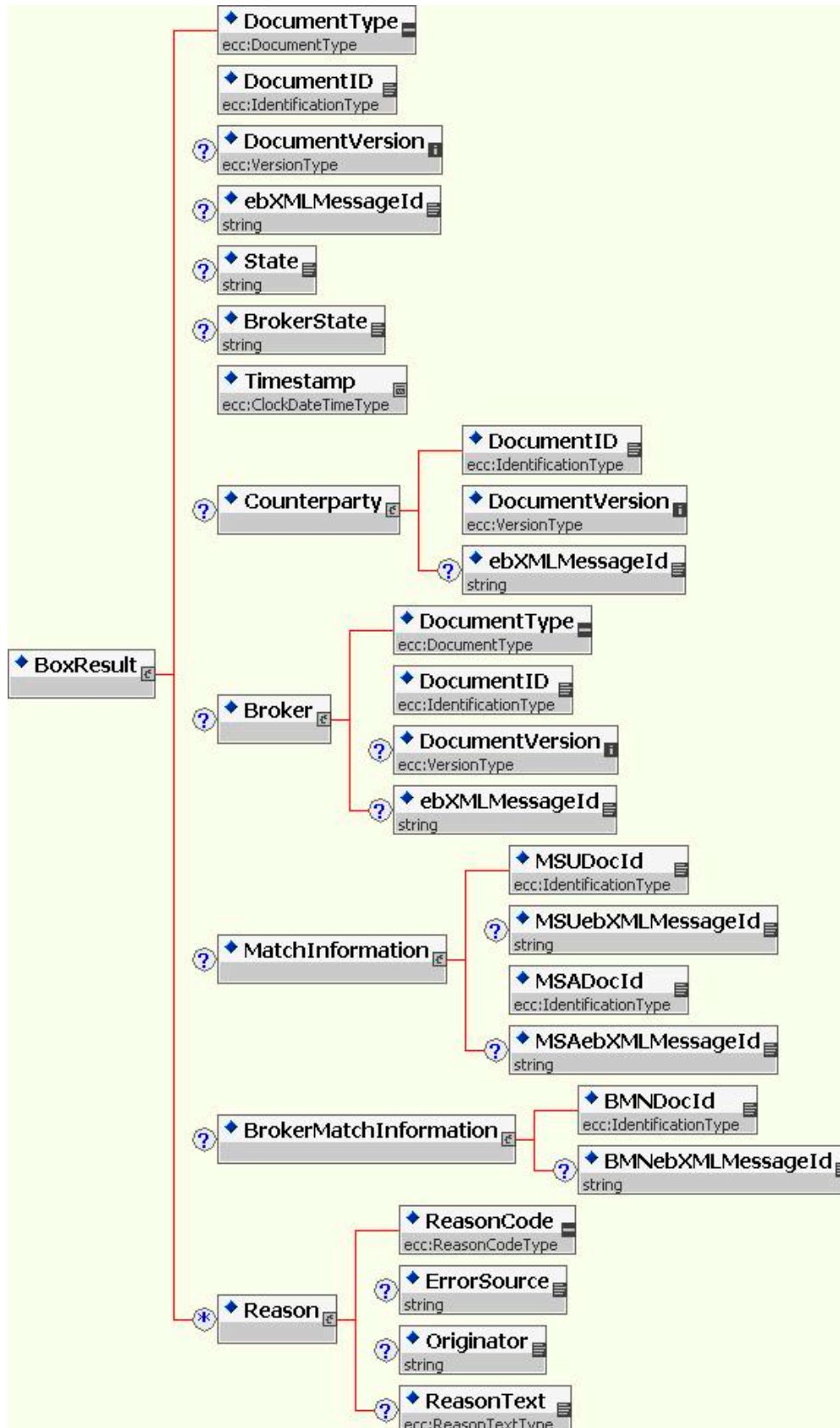


Figure 3: XML Schema for the BoxResult Document

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:lang="EN" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ecc="http://www.efet.org/ecm/schemas/v3r0/EFET-CoreCmpts-V3R0.xsd"
  elementFormDefault="qualified">
  <xsd:import
    namespace="http://www.efet.org/ecm/schemas/v3r0/EFET-
    CoreCmpts-V3R0.xsd"
    schemaLocation="EFET-CoreCmpts-V3R0.xsd"
  <xsd:include schemaLocation="EFET-CoreElements-V3R0.xsd"
  <xsd:element name="BoxResult">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="DocumentType" type="
        "ecc:DocumentType">
          <xsd:annotation>
            <xsd:documentation>The Document Type
            (i.e. CNF, BFI or
            CAN). </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="DocumentID" type="
        "ecc:IdentificationType">
          <xsd:annotation>
            <xsd:documentation>Unique
            identification of a
            document. </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="DocumentVersion" type="
        "ecc:VersionType" minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>Version of the
            document being sent. A document may be
            sent several times with the same
            identification. The version is used to
            distinguish one instance of the same
            document from another with the same
            identification. </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="ebXMLMessageId"
        type="xsd:string" minOccurs="0"
        <xsd:element name="State" type="xsd:string"
        minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>End State of the
            referenced
            Document. </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="BrokerState" type="xsd:string"
        minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>Broker End State of
            the referenced
            Document. </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="Timestamp"
        type="ecc:ClockDateTimeType">
          <xsd:annotation>
            <xsd:documentation>The Date and Time
            when the referenced Document reached
            the EndState. </xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="Counterparty" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="DocumentID"
              type="ecc:IdentificationType"

```

```

        <xsd:element name =
            "DocumentVersion"
            type="ecc:VersionType"
        >
            <xsd:element name =
                "ebXMLMessageId" type="xsd:string"
                minOccurs="0"
            >
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Broker" minOccurs = "0">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name = "DocumentType"
                    type = "ecc:DocumentType">
                        <xsd:annotation>
                            <xsd:documentation>The
                                Document Type of
                                the Broker Doc (i.e.
                                CNF, BFI).
                            </xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                <xsd:element name = "DocumentID"
                    type="ecc:IdentificationType"
                >
                <xsd:element name =
                    "DocumentVersion"
                    type="ecc:VersionType"
                    minOccurs="0"
                >
                <xsd:element name =
                    "ebXMLMessageId" type="xsd:string"
                    minOccurs="0"
                >
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="MatchInformation" minOccurs =
            "0">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name = "MSUDocId"
                        type="ecc:IdentificationType"
                    >
                    <xsd:element name =
                        "MSUebXMLMessageId"
                        type="xsd:string" minOccurs="0"
                    >
                    <xsd:element name = "MSADocId"
                        type="ecc:IdentificationType"
                    >
                    <xsd:element name =
                        "MSAebXMLMessageId"
                        type="xsd:string" minOccurs="0"
                    >
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="BrokerMatchInformation"
                minOccurs = "0">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name = "BMNDocId"
                            type="ecc:IdentificationType" />
                        <xsd:element name =
                            "BMNebXMLMessageId"
                            type="xsd:string" minOccurs="0" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element ref = "Reason" minOccurs="0"
                maxOccurs= "unbounded"
            >
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>

```

## **8 Valid Values and Enumerations**

Please refer to the EFET website for more information on valid values and enumerations when generating the XML files : <http://www.efet.org/ecm/staticdata/default.asp>