October 2005

# Reliable Messaging between SAP XI 3.0 and Microsoft BizTalk Server 2004 Using SOAP-Compliant Communication

André Fischer, Project Manager CTSC, SAP AG
Matthias Allgaier, Consultant XI, SAP Deutschland AG & Co. KG
Jürgen Daiberl, SAP Program Manager CTSC, Microsoft Corporation

## Summary

Reliable messaging is critical for web services. Many business problems cannot be solved unless the participants can be sure of the completion of message exchanges. Synchronous communication is not always an option due to its limitations concerning availability of the participants and performance issues. Reliable messaging is therefore a key feature needed for asynchronous communication using web services.

In May 2004, SAP and Microsoft announced a collaborative effort to work on SAP NetWeaver and Microsoft .NET interoperability – based on advanced web services that will allow, among other things, reliable messaging and adapter-less integration between Microsoft BizTalk and SAP Exchange Infrastructure (SAP XI). These features will be available in upcoming major releases of SAP NetWeaver and Microsoft .NET.

However, there are customers requesting a solution based on existing technologies. This whitepaper outlines how reliable messaging using SOAP-compliant communication between SAP XI 3.0 and Microsoft BizTalk Server 2004 can be achieved, fulfilling the Quality of Service "exactly once” and leveraging features and functions of the existing products.

## Applies to

- SAP Exchange Infrastructure 3.0 SP14
- Microsoft BizTalk Server 2004 SP1

## Keywords

SAP Exchange Infrastructure, Microsoft BizTalk Server, SOAP, HTTP, reliable messaging

## Level of Difficulty

Technical consultants, architects, developers, IT managers

## Contact

This document is provided to you by the Collaboration Technology Support Center Microsoft, a joint team from SAP and Microsoft that drives interoperability. For feedback or questions you can contact the CTSC at ctsc@sap.com or ctsc@microsoft.com. Please check the .NET interoperability area in the SAP Developer Network (http://sdn.sap.com) for any updates or further information.

## Contents

## Table of Figures

## Introduction

Reliable messaging is critical for web services. Many business problems cannot be solved unless the participants can be sure of the completion of message exchanges. One can distinguish two types of web services communication: *synchronous* and *asynchronous* communication.

Synchronous communication is not always an option due to its limitations concerning availability of the participants and performance issues. In the case of synchronous communication, the web service client must suspend its processing until the web service that has been called completes processing the request and returns a response. As a result, both parties – the sender and the receiver – have to be available. The receiving system might not be available for several reasons. The server itself can be down or the communication lines might not be available. As a result, the business process is blocked since it has to wait for an answer from the business partner.

If the calling component performs an asynchronous call instead, the calling system can proceed with its task without waiting for the receiving component to do its job. If asynchronous communication using web services is chosen, reliable messaging is a must.

In May 2004, SAP and Microsoft announced a collaborative effort to work on interoperability between SAP NetWeaver and Microsoft .NET based on advanced web services that would allow reliable messaging between Microsoft BizTalk and SAP XI infrastructure. These features will be available with upcoming major releases of SAP NetWeaver and Microsoft .NET.

However, there are customers requesting a solution based on existing technologies. Therefore, we investigated how it is possible to achieve reliable messaging using existing web services technology and existing features and functions of SAP XI 3.0 and Microsoft BizTalk Server 2004. We have been able to setup reliable messaging between SAP XI 3.0 and Microsoft BizTalk Server 2004 using SOAP-compliant communication that fulfils the Quality of Service *"exactly once".* Since both products use different SOAP 1.1 extension header elements and adapters to achieve reliable messaging in SOAP-compliant communication, we had to use different setups for each direction. While for the communication from SAP XI to Microsoft BizTalk 2004 one has to rely on the BizTalk Server Framework 2.0, the communication from Microsoft BizTalk 2004 to SAP XI is using the SAP XI SOAP header.

The "How-to Guide" section of this document does not provide the complete details on this configuration, but it does concentrate on the stumbling blocks we found during our investigation.

## Reliable Messaging

### What is Reliable Messaging?

The concept of reliable messaging is the act of sending one or more messages from a sender to a receiver in a manner that guarantees a level of assurance that message(s) are delivered. Thereby the message delivery service has some quality, the so-called "Quality of Service" (QoS) that can be requested unilaterally by the sender by populating message elements designed for this purpose. To achieve this goal for the web services world, an interoperable reliable messaging specification is needed as a standard that is supported by all vendors.

### About to Come: Enhanced Web Services

Since the need for web services reliable messaging was apparent to everyone, an attempt was made at developing a common standard for enhanced web services. Last year, Microsoft and SAP agreed to expand their cooperation around defining and ensuring interoperability of advanced web services and implement them in their respective products. WS-ReliableMessaging specifications have been submitted to the newly-formed work group, "*OASIS Web Services Reliable Exchange (WS-RX) TC*." With the next major releases of Microsoft and SAP NetWeaver, an adapter-less communication based on enhanced web services will be possible between SAP XI and Microsoft BizTalk.

This whitepaper describes an integration scenario to use in the meantime.

### Reliable Messaging Using SAP XI 3.0 and Microsoft BizTalk 2004

In the past, well-adopted standards for reliable messaging on top of the SOAP message format did not exist. The only possibility to address reliable messaging in the context of web services for any vendor was therefore to fall back on proprietary mechanisms. Common to the approaches used in SAP XI and Microsoft BizTalk Server is the use of proprietary SOAP 1.1 extension header elements. SAP introduced SAP XI-specific SOAP extension header elements to achieve this goal, while Microsoft introduced the BizTalk Framework 2.0. To achieve reliable messaging using SOAP-compliant communications between SAP XI 3.0 and Microsoft BizTalk Server 2004 using standard interfaces, we took the following approach:

Reliable messaging can be achieved out-of-the-box if the sending system is sending a message in a format that complies with the SOAP header extensions of the receiving system. Both products support the receipt of XML documents from third-party components with a specified Quality of Service (QoS), unless the sending party sends the XML documents compliant to its specifications.

As a result we chose:

- BizTalk Server Framework 2.0 to achieve reliable messaging from SAP XI to Microsoft BizTalk Server;
- SAP XI SOAP header extensions to achieve reliable messaging from Microsoft BizTalk Server to SAP XI.

The architecture for these scenarios is described in the next two sections of this collaboration brief.

## Architecture: Reliable Messaging from XI to BizTalk

In order to guarantee reliable messaging, BizTalk Server 2004 provides the BizTalk Server Framework 2.0 (BTF2). BTF2 is based on a request/response processing model using specific SOAP 1.1 header extensions to transfer in formations used to guarantee reliable messaging.

In our solution we have developed an XI Adapter Framework module for the SOAP Adapter that transforms a XI message into a BTF2 message using the XI MessageID as the unique identifier within the BTF2 message. The message is posted to BizTalk using the standard XI SOAP Adapter. In BizTalk the message is received and a duplicate message check is performed within the BTF2 Framework.

If necessary BizTalk can send back a BTF2 acknowledgment to the Plain HTTP Adapter of XI (optional). In this collaboration brief we will present an extended scenario including a resend mechanism in XI using the Business Process Engine (BPE).

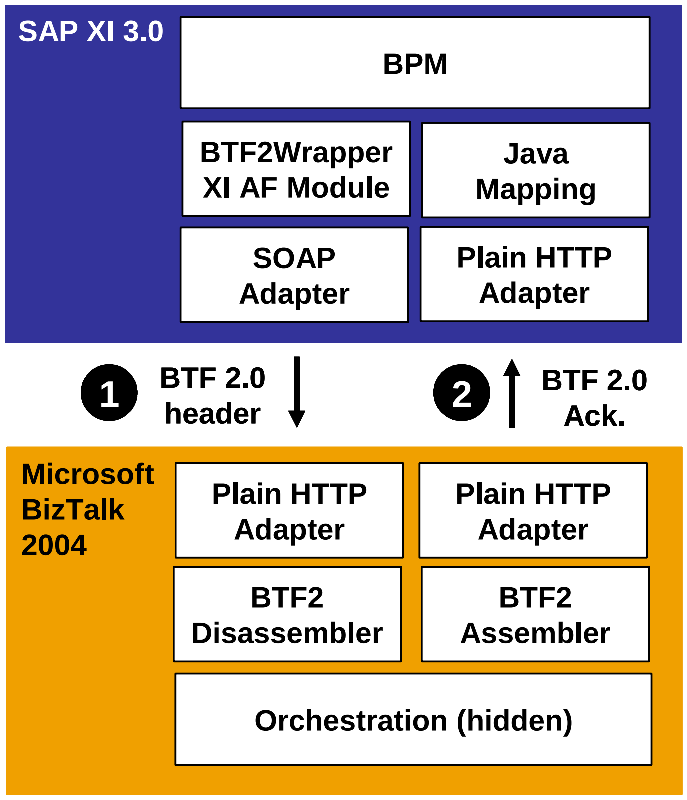


**Figure 1 Architecture: Reliable Messaging from XI to BizTalk**

## Architecture: Reliable Messaging from BizTalk to XI

The XI SOAP Adapter supports Exactly-Once (EO) end-to-end processing by being called with a unique MessageID that can be specified as part of the query string in the URL. We use this feature and call the published web service on XI with a URL that contains the BizTalk message ID as the unique identifier.

The BizTalk message ID is assigned to a string that is added as an additional parameter to the URL that is called by the dynamic port.

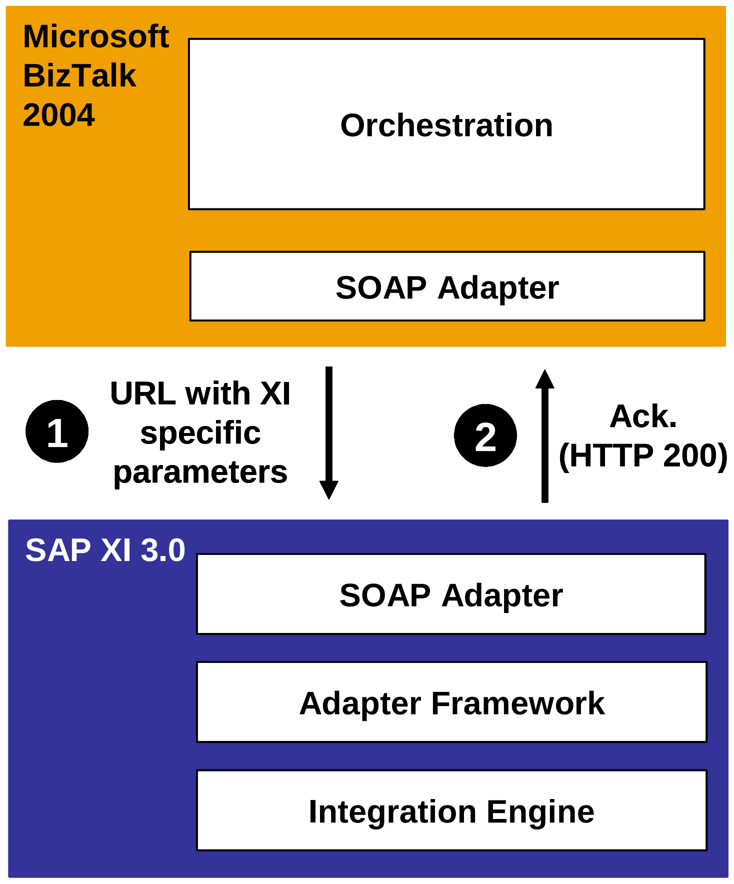


**Figure 2 Architecture: Reliable Messaging from BizTalk to XI**

## Microsoft BizTalk Server

### Reliable Messaging Using Microsoft BizTalk Framework 2.0

The BizTalk Server 2004 supports reliable messaging using the Microsoft BizTalk Server Framework 2.0 (BTF2). A BizTalk Framework 2.0-compliant document consists of a standard SOAP 1.1 message that contains BizTalk-specific header entries, such as a unique GUID.

The BizTalk Framework Disassembler pipeline component parses XML data and determines whether it contains a BizTalk Framework-based messaging payload.

If the sender requested an acknowledgment for the generated message using the *deliverReceiptRequest* header within the BizTalk Framework envelope, it will be generated by the BizTalk Framework Disassembler pipeline component.

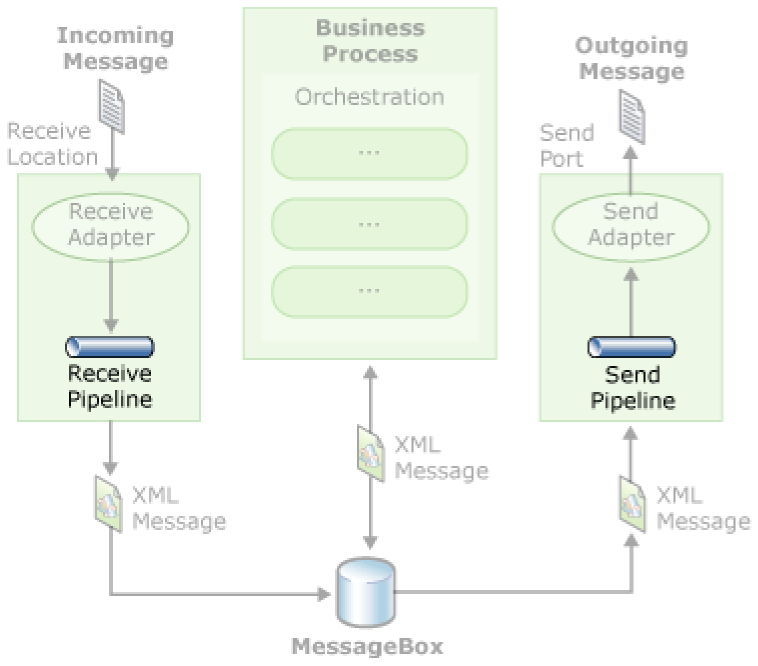


**Figure 3 BizTalk: Send and Receive Pipelines**

When using HTTP as the communication protocol, the HTTP Adapter from BizTalk Server has to be used.

## Microsoft BizTalk Framework 2.0 Document Structure

A BizTalk Framework 2.0-compliant document consists of a standard SOAP 1.1 message that contains the following:

- An application-specific Business Document (in this case a simple XML document containing just one entry), with its own application-defined XML namespace, carried in the body of the SOAP message.
- BizTalk-specific SOAP header entries

The following is an example of a simple BizTalk Document that was used in our demo scenario:

```
<?xml version="1.0" encoding="UTF8" ?>
 <SOAPENV:Envelope xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchemainstance">
        <SOAPENV:Header>
            <eps:endpoints
                  xmlns:eps="http://schemas.biztalk.org/btf20/endpoints"
                  SOAPENV:mustUnderstand="1"
                  xmlns:biz="http://schemas.biztalk.org/btf20/address/types">
            <eps:to>
              <eps:address xsi:type="biz:OrganizationName">Receiver</eps:address>
            </eps:to>
           <eps:from>
             <eps:address xsi:type="biz:OrganizationName">Sender</eps:address>
          </eps:from>
          </eps:endpoints>
          <prop:properties xmlns:prop="http://schemas.biztalk.org/btf20/properties"
                   SOAPENV:mustUnderstand="1">
            <prop:identity>uuid:8D002C50A3413A48B979CFADF15CDBB2</prop:identity>
            <prop:sentAt>20051020T11:04:21+00:00</prop:sentAt>
            <prop:expiresAt>20051231T15:13:43+00:00</prop:expiresAt>
            <prop:topic>root:Root</prop:topic>
          </prop:properties>
          <services:services
               xmlns:services="http://schemas.biztalk.org/btf20/services">
              <services:deliveryReceiptRequest>
                <services:sendTo>
                  <services:address
                    xsi:type="biz:httpURL">http://p136796.wdf.sap.corp:8000/sap/xi/
                    adapter_plain?namespace=http%3A//msctsc.sap.corp/xi2biz&
                    interface=BTF2_AckMessage_Outbound&service=MSCTSC_BizTalk_
                    2004&party=&agency=&scheme=&QOS=EO&sapuser=xiappluser
                    &sappassword=passpass&sapclient=100
                </services:address>
               </services:sendTo>
              <services:sendBy>20051231T15:13:43+00:00</services:sendBy>
            </services:deliveryReceiptRequest>
          </services:services>
</SOAPENV:Header>
<SOAPENV:Body>
          <ns0:Root xmlns:ns0="http://BTF2Test1.Schema1">
                    <test>SDN Article</test>
          </ns0:Root>
</SOAPENV:Body>
</SOAPENV:Envelope>
```

**Figure 4 Example: BizTalk Server Framework 2.0 Document**

## Microsoft BizTalk Framework 2.0 Header Entries

From the BizTalk-specific SOAP header entries we need the following to achieve reliable messaging since they are marked as mandatory:

- endpoints
- properties
- services

The following SOAP header entries are only required if the sender wants to receive an acknowledgement.

*<endpoints>*

Though the <endpoints> header entry is marked as mandatory (which is reflected by the usage of the SOAP-ENV: mustUnderstand="1" attribute) the elements can contain dummy values if a simple example is used as in our setup.

The <to> and <from> tags contain the specification of the source and destination business entity that can be used to create internal routing rules. In our setup we used dummy values here as well.

*<properties>*

Document identity information and other properties are specified by a SOAP header entry marked by the <properties> tag.

- Identity
- sentAt
- expiresAt
- topic

Important for the setup of our demo scenario are only the first three tags so that we filled the <topic> tag with a dummy value.

The <identity> tag must contain a GUID that is used as the message ID of the BizTalk message after the receipt of the BTF2 document.

In our demo scenario the GUID was filled with the message ID of the SAP XI message that started the Business Process in XI thus allowing having a direct link between the XI and BizTalk message.

The <sentAt> tag must contain the timestamp when the document has been sent to BizTalk.

The <expiresAt> tag must contain expiration timestamp of the Document. Beyond the point in time stamped in this element, the associated BizTalk Document is considered to have expired and must not be processed or acknowledged by the BizTalk Server.

*<services>*

Reliable delivery services for a BizTalk document are specified by an optional SOAP header entry marked by the <services> tag. If omitted BizTalk guarantees a duplicate check if the message is sent several times but does not send an acknowledgement to the sender.

- sendTo
- sendBy

If the <deliveryReceiptRequest> element is present BizTalk, is required to send a delivery receipt back to the address given in the <sendTo> sub-element upon receiving and accepting the BizTalk Document. In the following two examples are given. One for a file and the other for an HTTP based receiving address:

```
<services:address
          xsi:type=" biz:fileURL "> file://C:\XIBizTalk\BTF2Receipt\Ack\btf-
    ack-to-message-%MessageId%.xml
</services:address>
```

**Figure 5  BTF2: <deliveryReceiptRequest> - File-based Receiving Address**

```
<services:address
            xsi:type="biz:httpURL">http://p136796.wdf.sap.corp:8000/sap/xi/
            adapter_plain?namespace=http%3A//msctsc.sap.corp/xi2biz&
            interface=BTF2_AckMessage_Outbound&service=MSCTSC_BizTalk_
            2004&party=&agency=&scheme=&QOS=EO
</services:address>
```

**Figure 6 BTF2: <deliveryReceiptRequest> - HTTP-based Receiving Address**

The <sendBy> tag contains a timestamp until this point in time the sending application expects to receive an acknowledgement by the BizTalk Server about receiving and accepting the document.

## SAP XI

### Reliable Messaging Using the XI Messaging Protocol

The messaging protocol used by SAP XI also relies on SOAP message format. An XI Messaging Protocol compliant document consists of a standard SOAP 1.1 message that contains XI specific header entries such as a unique GUID.

There are two options for a SOAP client to provide a unique GUID for SAP XI using the SOAP Sender Adapter:

- The first option is to use a valid XI message header in the SOAP message header.
- The second option is to specify the GUID and other values in a corresponding query string in the URL.

If the SOAP client then receives an empty SOAP message in HTTP 200 as a response, this means that the message has been successfully persisted and will be processed asynchronously exactly once.

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP:Header />
        <SOAP:Body />
</SOAP:Envelope>
```

**Figure 7 SAP XI: HTTP Response of SOAP Adapter**

If the SOAP client receives a SOAP fault or any response not in HTTP 200, it must resend the message until it receives an empty SOAP message in HTTP 200. The sender of the message uses the attribute *Quality of Service* (QoS) to determine how a message is delivered.

### SAP XI SOAP Document Structure

A SAP XI SOAP compliant document consists of a standard SOAP 1.1 message that contains the following:

- An application-specific Business Document (in this case a simple XML document containing just one entry), with its own application-defined XML namespace, carried in the body of the SOAP message as an attachment.
- SAP XI -specific SOAP header entries

From the SAP XI-specific SOAP header entries we need the following to achieve reliable messaging:

- Main
- ReliableMessaging

Instead of using a complete SAP XI SOAP Header the parameters contained in this header can be passed to the SAP XI SOAP adapter using a query string. To do so one has to configure the *Conversion Parameters* in the communication channel such that the option *Use Query String* is checked.

If the SOAP adapter is configured as shown in the figure above it is possible to pass the XI SOAP header elements as parameters using the following syntax:

```
http://host:port/XISOAPAdapter/MessageServlet?channel=party:service:channel&MessageId=<GUID>&version=3.0
```

Passing the parameters of the example mentioned above it would be possible to send the payload of the XML message using the following URL parameters:

http://p136796:50000/XISOAPAdapter/MessageServlet?channel=:MSCTSC_BizTalk_2004:SOAP_Sender_BizTalk&MessageId=8F02C3E0-C2CD-4B6B-AE07-D95B2275834E&version=3.0

An example of an XML message containing SAP XI SOAP header is shown in the following section:

```
 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
                 xmlns:SAP="http://sap.com/xi/XI/Message/30">
 <SOAP:Header>
  <SAP:Main xmlns:SAP="http://sap.com/xi/XI/Message/30"
         xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:wsu="http://www.docs.oasis-open.org/wss/2004/01/oasis-200401-
         wss-wssecurity-utility-1.0.xsd" versionMajor="003" versionMinor="000"
        SOAP:mustUnderstand="1" wsu:Id="wsuid-main-
         92ABE13F5C59AB7FE10000000A1551F7">
  <SAP:MessageClass>ApplicationMessage</SAP:MessageClass>
 <SAP:ProcessingMode>asynchronous</SAP:ProcessingMode>
 <SAP:MessageId>8F02C3E0-C2CD-4B6B-AE07-D95B2275834E</SAP:MessageId>
 <SAP:TimeSent>2005-10-23T22:10:50Z</SAP:TimeSent>
  <SAP:Sender>
          <SAP:Service>MSCTSC_BizTalk_2004</SAP:Service>
         <SAP:Interface  namespace="http://msctsc.sap.corp/xi">
         MIF_XI_SOAP_EO_Outbound</SAP:Interface>
 </SAP:Sender>

<SAP:Interface namespace="http://msctsc.sap.corp/xi">
         MIF_XI_SOAP_EO_Outbound</SAP:Interface>
</SAP:Main>
<SAP:ReliableMessaging xmlns:SAP="http://sap.com/xi/XI/Message/30"
        xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP:mustUnderstand="1">
         <SAP:QualityOfService>ExactlyOnce</SAP:QualityOfService>
</SAP:ReliableMessaging>
</SOAP:Header>
 <SOAP:Body>
          <ns:MT_XI_SOAP_EO xmlns:ns="http://msctsc.sap.corp/xi"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
         xmlns:SAP="http://sap.com/xi/XI/Message/30"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <field1 xsi:type="xsd:string">Value x</field1>
         <field2 xsi:type="xsd:string">Value y</field2>
         </ns:MT_XI_SOAP_EO>
 </SOAP:Body>
</SOAP:Envelope>
```

**Figure 8 Example: SAP XI Message Containing SAP XI SOAP Header**

## SAP XI SOAP Header Entries

From the SAP XI-specific SOAP header entries we need the following to achieve reliable messaging since they are marked as mandatory:

- Main
- ReliableMessaging

However not all sub tags have to be filled if a URL is used that passes the appropriate parameters.

*<Main>*

The MessageClass and ProcessingMode tags will be filled automatically by the adapter engine.

The tag MessageID will be filled with the value that is passed using the parameter MessageID in the URL.

The tag timeSent is again automatically filled by the adapter engine.

The parameters Service and Interface are filled using the parameters passed by the calling URL.

*<ReliableMessaging>*

Reliable delivery services for a XI document are specified by the mandatory SOAP header entry marked by the <ReliableMessaging> tag.

The <QualityOfService> tag contains the Quality of Service requested by the SOAP client.

Since we are passing parameters using a URL this tag is filled by the adapter engine depending on the configuration settings of the XI interface.

## Reliable Messaging from XI to BizTalk

### Overview

In our setup we used BizTalk Server Framework 2.0 to achieve reliable messaging from SAP XI to Microsoft BizTalk Server. Therefore it is necessary to post a BTF2 compliant message to the Plain HTTP-Adapter of BizTalk using the BTF2 Disassembler. With the usage of BTF2 BizTalk guarantees that the same XI message (identified by the UUID in the BTF2 SOAP Envelope) is not processed twice. Furthermore the BTF2 Framework also provides functionality to sent back acknowledgements to the sending application (XI). The sender can specify a HTTP URL where BTF2 should send the acknowledgement to. Based on the BTF2 acknowledgements the sending application can implement a resend mechanism (if required).

In general there are two use cases to setup reliable messaging between SAP XI and MS BizTalk:

1. XI/BizTalk integration including a resend mechanism in XI (ccBPM)
   In this scenario XI will create a BTF2 compliant SOAP message and requesting a BTF2 acknowledgement from BizTalk. If the BTF2 acknowledgement is not received within a specified period of time XI will resend the same message again to BizTalk.
2. XI/BizTalk integration without a resend mechanism in XI
   It is also possible that XI creates a BTF2 compliant SOAP message without requesting BizTalk to send a BTF2 acknowledgement. As in use case 1 the BTF2 Framework will provide duplicate message checks on BizTalk. We recommend applying this use case in high message volume scenarios because there is no overhead of BTF2 acknowledgement messages.

The developed SOAP adapter module supports both use cases. With the configuration parameter “Request_BTF2Ack” you can choose whether you want to use the module in context of scenario type 1 (true) or 2 (false).

## Message Flow between XI and BizTalk

In this chapter we outline the necessary steps to setup XI/BizTalk integration including a resend mechanism in XI (Use case 1). Beside the configuration of the BPM process and the adapter module we will focus on the SOAP messages exchanged by XI and BizTalk.

The following steps are required:

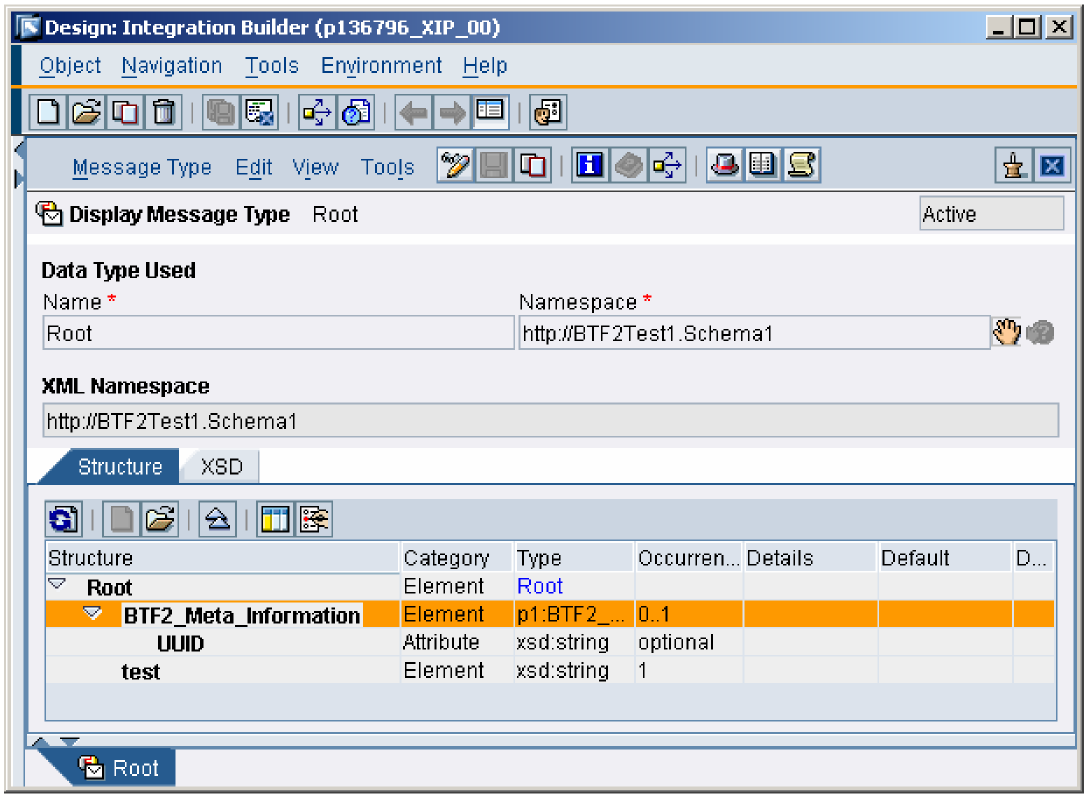
1. A client application triggers a BPM process to send a message to BizTalk. Since the BPM process is responsible to resend the message to BizTalk in case of no acknowledgement was received in the specified period of time a unique MessageID is required. Within a BPM process instance it is not possible to read the MessageID of a received message stored in a container element. Therefore the MessageID of the message that triggers the BPM process is used. The MessageID is extracted using a Message Mapping and written to payload of the message.
2. BPM process is started and a correlation based on the inbound MessageID is created and activated
3. BPM process sends the message to BizTalk using a SOAP-Receiver Adapter. The BizTalk specific BTF2 SOAP format is created within an own developed XI AF module that is called by the module processor of the XI Adapter Framework.
4. BizTalk receives the message and creates a BTF2 acknowledgement message that is sent back to the Plain HTTP-Adapter of XI.
5. In XI the Plain HTTP-Adapter receives the message and routes it to the appropriate BPM process instance based on the UUID. Within BPM a receive step receives the message using the correlation defined in step 2. Afterwards the BPM process is completed.
6. If the BPM process does not receive an acknowledgement within one second it resends the message again to BizTalk. If again no response is received from BizTalk the BPM process is cancelled.

In the following the design, configuration and the SOAP message exchange between XI and BizTalk are described.

## Setup of XI Part

### Design I: Message Type & Mappings

As described in the last section the BPM process uses the MessageID of the inbound message that triggers the process as the unique identifier. To hand over the MessageID to the BPM process the BizTalk message type “Root” has to be extended with the XML tag and attribute:  <BTF2_Meta_Information UUID=""/>


Design: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Message Type Edit View Tools

Display Message Type Root — Active

**Data Type Used**

| Name * | Namespace * |
|---|---|
| Root | http://BTF2Test1.Schema1 |

**XML Namespace**: http://BTF2Test1.Schema1

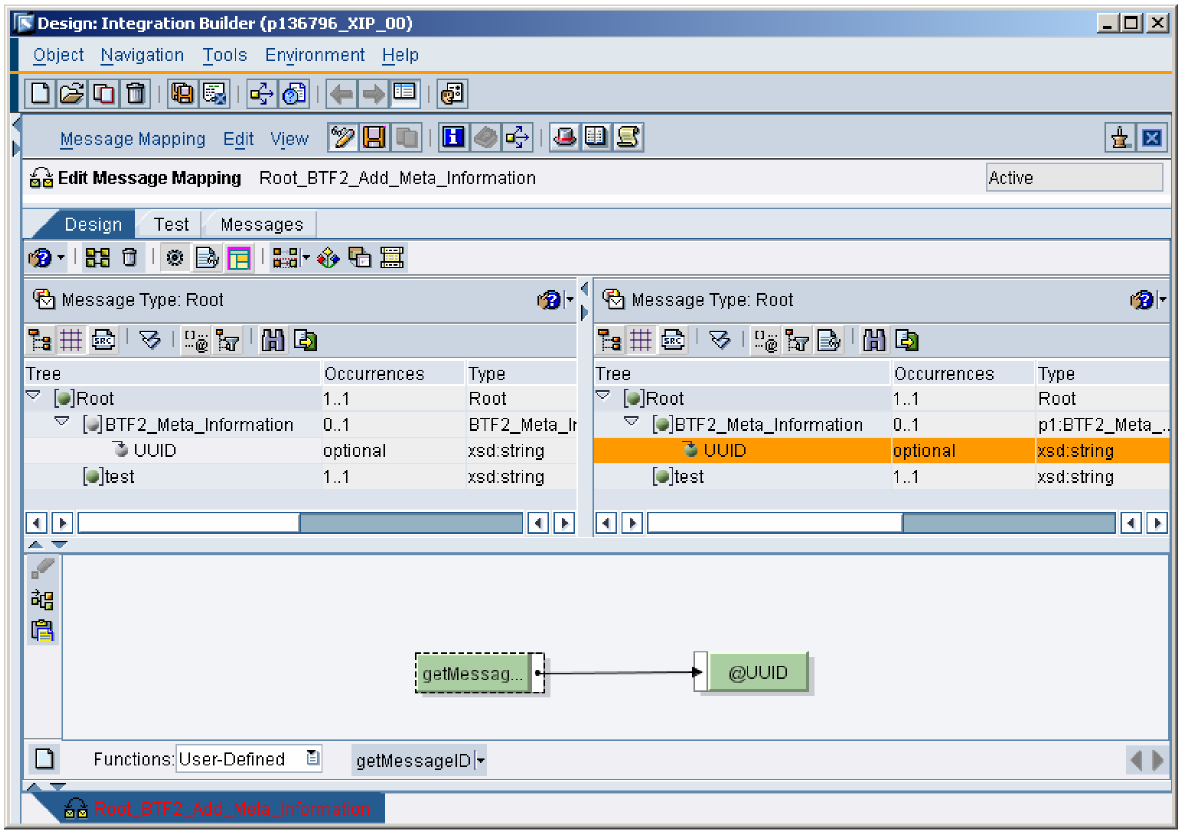
Structure | XSD

| Structure | Category | Type | Occurren... | Details | Default | D... |
|---|---|---|---|---|---|---|
| Root | Element | Root | | | | |
| BTF2_Meta_Information | Element | p1:BTF2_... | 0..1 | | | |
| UUID | Attribute | xsd:string | optional | | | |
| test | Element | xsd:string | 1 | | | |

Root

**Figure 9  SAP XI: Message Type Definition**

The name of the XML tag and attribute to store the MessageID has to be exactly “BTF2_Meta_Information” and “UUID”. This tag will be used by the SOAP adapter module to read the MessageID.  To read the MessageID and assign it to the “UUID” attribute the following Message Mapping is used:

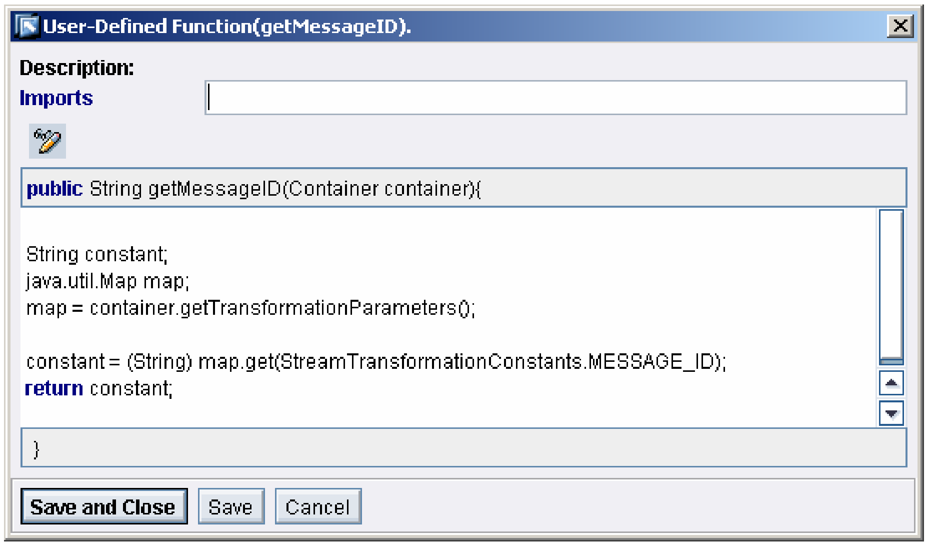

Design: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Message Mapping Edit View
Edit Message Mapping Root_BTF2_Add_Meta_Information Active
Design Test Messages
Message Type: Root
getMessag...
@UUID
Functions: User-Defined
getMessageID
Root_BTF2_Add_Meta_Information

| Tree | Occurrences | Type |
|---|---|---|
| Root | 1..1 | Root |
| BTF2_Meta_Information | 0..1 | BTF2_Meta_I |
| UUID | optional | xsd:string |
| test | 1..1 | xsd:string |

| Tree | Occurrences | Type |
|---|---|---|
| Root | 1..1 | Root |
| BTF2_Meta_Information | 0..1 | p1:BTF2_Meta_.. |
| UUID | optional | xsd:string |
| test | 1..1 | xsd:string |

**Figure 10  SAP XI: Message Mapping**

The MessageID is read by a user-defined Java function:

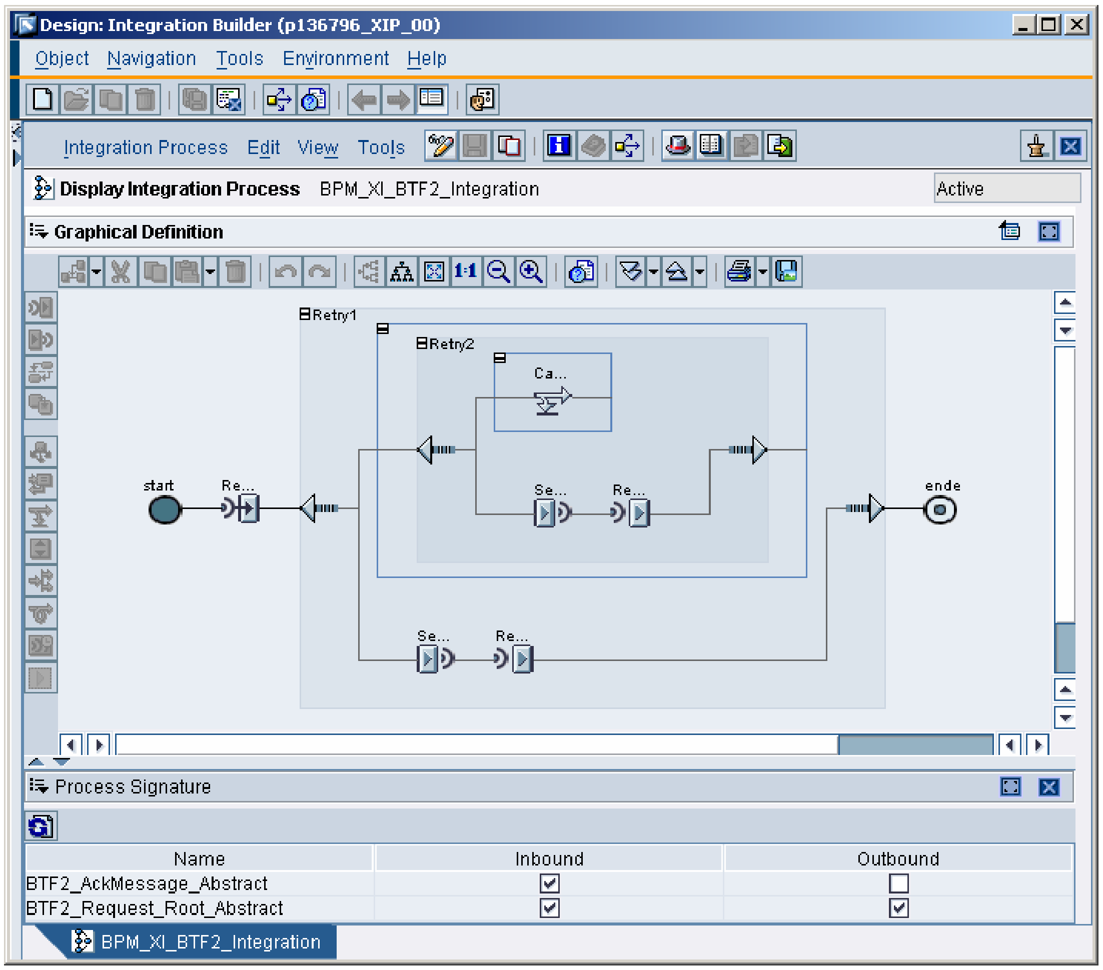


```
public String getMessageID(Container container){

String constant;
java.util.Map map;
map = container.getTransformationParameters();

constant = (String) map.get(StreamTransformationConstants.MESSAGE_ID);
return constant;

}
```

**Figure 11  SAP XI: Retrieval of MessageID**

## Design II: ccBPM – BPM_XI_BTF2_Integration

For demonstrating the “two step handshake” communication between XI and BizTalk the BPM process “BPM_XI_BTF2_Integration” is used.


Design: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Integration Process Edit View Tools
Graphical Definition
Retry1
Retry2
Ca...
start
Re...
Se...
Re...
ende
Se...
Re...
Process Signature
BPM_XI_BTF2_Integration

Display Integration Process BPM_XI_BTF2_Integration Active

| Name | Inbound | Outbound |
|---|---|---|
| BTF2_AckMessage_Abstract | ✓ | ☐ |
| BTF2_Request_Root_Abstract | ✓ | ✓ |

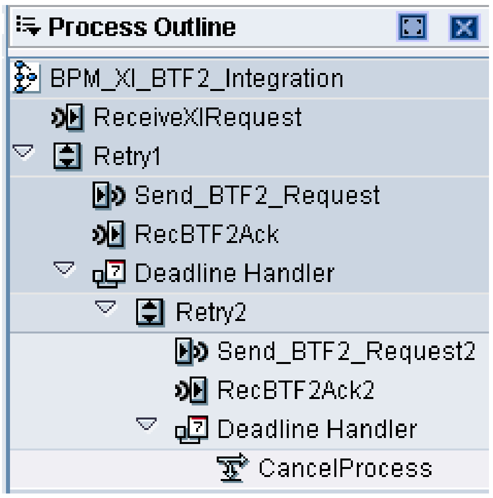
**Figure 12 SAP XI: BPM Process**

The BPM process is started by a message of type “BTF2_Request_Root_Abstract” (see process signature). The abstract message interface of the inbound message refers to the message type described in the previous section and the inbound message therefore contains in its payload a unique MessageID.

In the next step the same message is sent to BizTalk using the send step “Send_BTF2_Request”. Afterwards the control flow waits in a receive step (“RecBTF2Ack”) until a BTF2 acknowledgement of BizTalk is received. If the acknowledgement is not received within a specific timeframe the deadline branch of block “Retry1” is triggered and the same message as sent in the first send step (containing the same MessageID) is sent again to BizTalk using the send step “Send_BTF2_Request2”.

If the process again does not receive a BTF2 acknowledgement from BizTalk the deadline branch of block “Retry2” is triggered and the process instance is terminated using a control step. Note. It is also possible to raise an alert in this error situation.

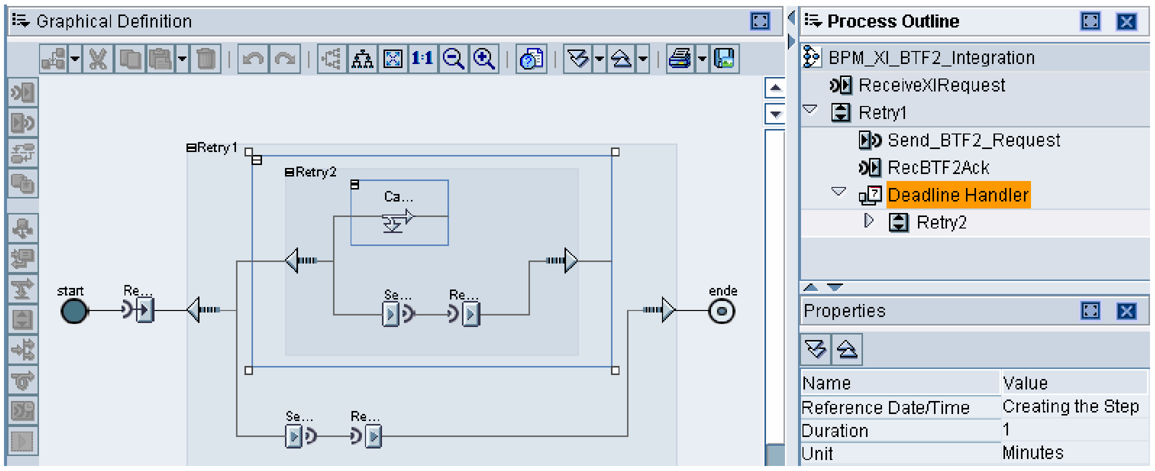
The process logic of the described BPM process is shown in the next screenshot as the process outline:


Process Outline

BPM_XI_BTF2_Integration
- ReceiveXIRequest
- Retry1
  - Send_BTF2_Request
  - RecBTF2Ack
  - Deadline Handler
    - Retry2
      - Send_BTF2_Request2
      - RecBTF2Ack2
      - Deadline Handler
        - CancelProcess

**Figure 13 BPM Process – Process Outline**

The next screenshot shows the details of the “Retry1” block indicating a timeframe of 1 minute before triggering the deadline branch:



Process Outline

BPM_XI_BTF2_Integration
- ReceiveXIRequest
- Retry1
  - Send_BTF2_Request
  - RecBTF2Ack
  - Deadline Handler
    - Retry2

Properties

| Name | Value |
|---|---|
| Reference Date/Time | Creating the Step |
| Duration | 1 |
| Unit | Minutes |

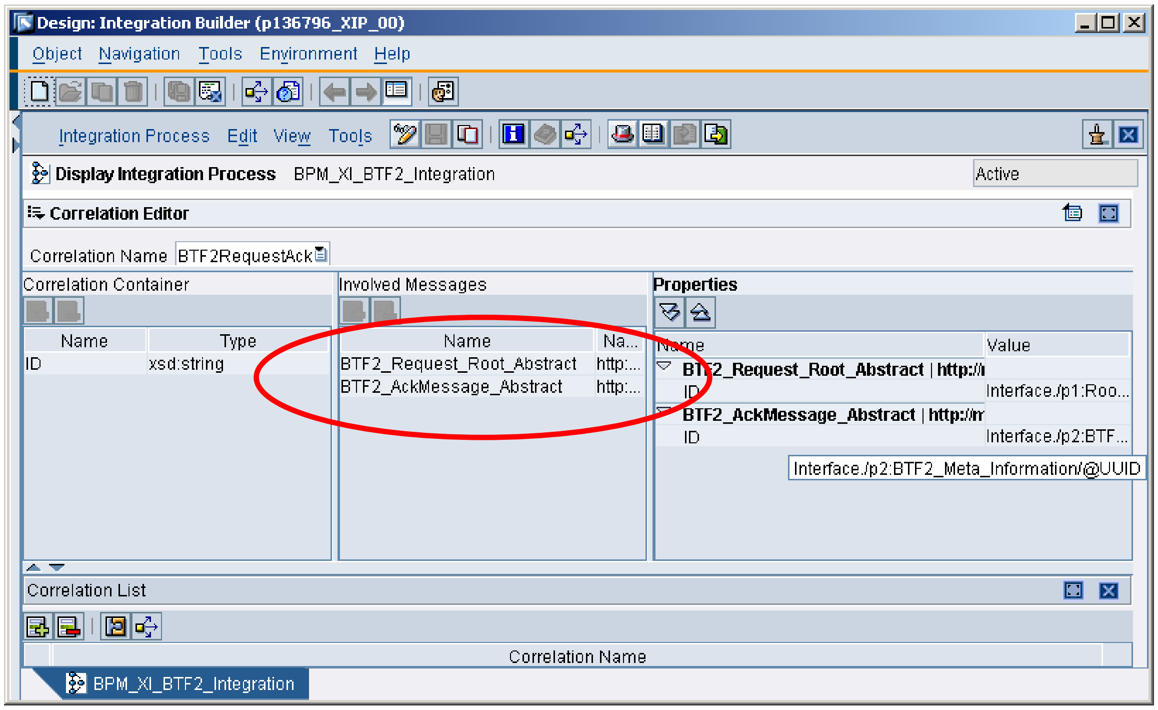
**Figure 14 BPM Process – Retry Block**

In order to implement the logic of sending a message to BizTalk and receiving the appropriate BTF2 acknowledgment back from BizTalk the correlation of these two messages is essential.

As implemented in our solution the message sent to BizTalk (instance of abstract message interface “BTF2_Request_Root_Abstract”) as well as the BTF2 acknowledgement receiving from BizTalk (instance of abstract message interface “BTF2AckMessage_Abstract”) both implement the meta information tag

“BTF2_Meta_Information” with the unique MessageID as the value of the attribute “UUID”. The correlation is activated in the first receive step of the BPM process.

Based on these common fields a correlation can be defined using the Correlation Editor:


Design: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Integration Process Edit View Tools
Correlation List
BPM_XI_BTF2_Integration

Display Integration Process BPM_XI_BTF2_Integration Active

Correlation Editor

Correlation Name BTF2RequestAck

Correlation Container

| Name | Type |
|---|---|
| ID | xsd:string |

Involved Messages

| Name | Na... |
|---|---|
| BTF2_Request_Root_Abstract | http:... |
| BTF2_AckMessage_Abstract | http:... |

Properties

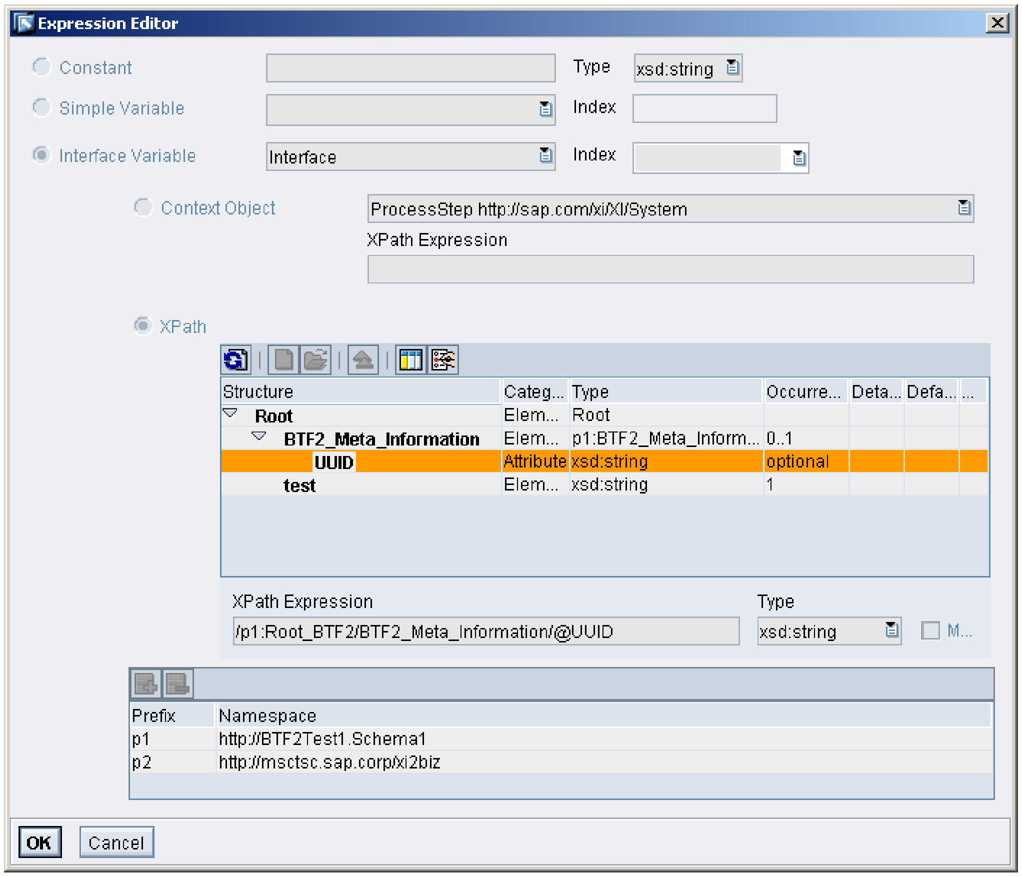
| Name | Value |
|---|---|
| **BTF2_Request_Root_Abstract \| http://r** | |
| ID | Interface./p1:Roo... |
| **BTF2_AckMessage_Abstract \| http://r** | |
| ID | Interface./p2:BTF... |

Interface./p2:BTF2_Meta_Information/@UUID

Correlation Name

**Figure 15 SAP XI - Correlation Editor**

The correlation is defined using an XPath expressions referring to the “UUID” attribute of the “BTF2_Meta_Information” tag:



Expression Editor

Constant — Type: xsd:string

Simple Variable — Index

Interface Variable [✓]: Interface — Index

Context Object: ProcessStep http://sap.com/xi/XI/System

XPath Expression

XPath [✓]

| Structure | Categ... | Type | Occurre... | Deta... | Defa... | ... |
|---|---|---|---|---|---|---|
| Root | Elem... | Root | | | | |
| BTF2_Meta_Information | Elem... | p1:BTF2_Meta_Inform... | 0..1 | | | |
| UUID | Attribute | xsd:string | optional | | | |
| test | Elem... | xsd:string | 1 | | | |

XPath Expression: /p1:Root_BTF2/BTF2_Meta_Information/@UUID

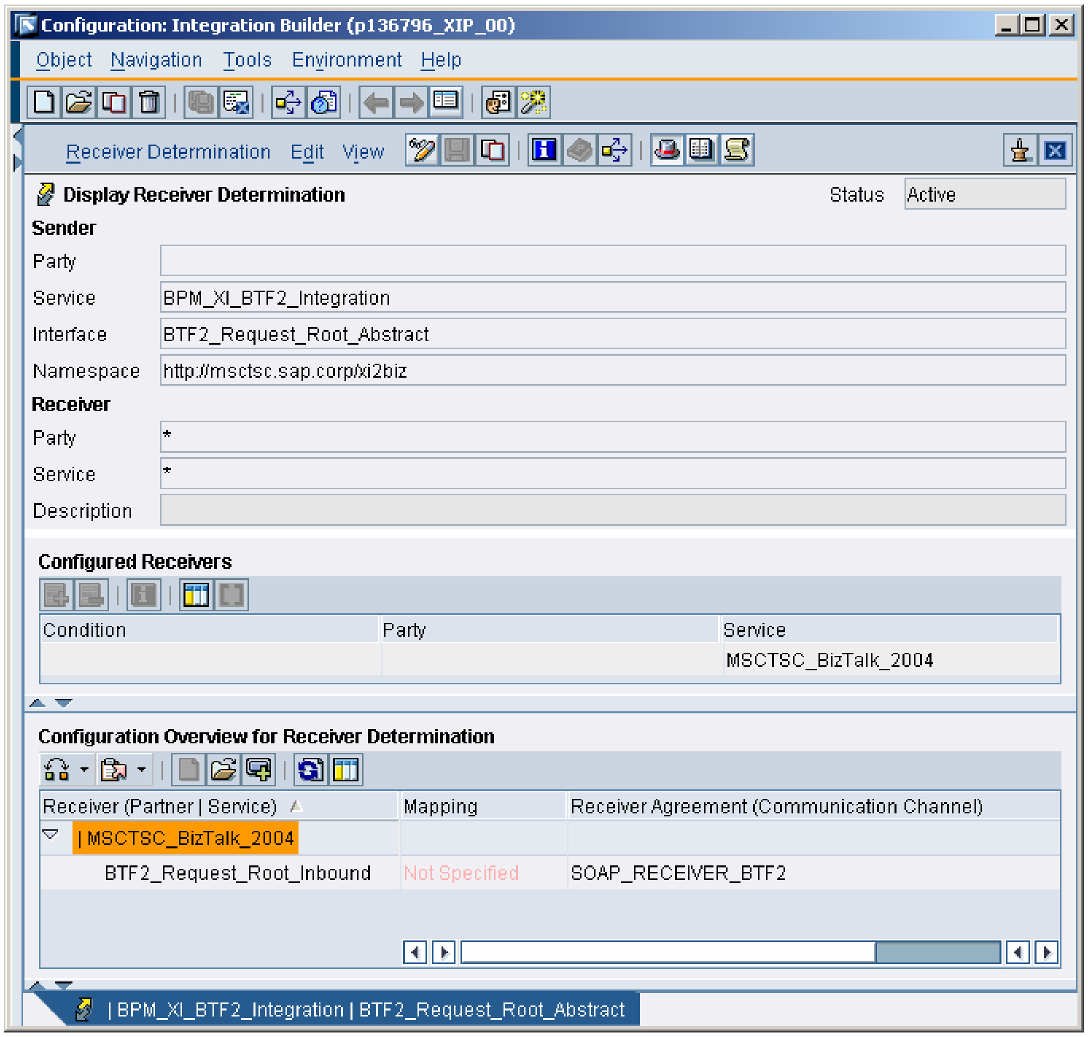
Type: xsd:string

| Prefix | Namespace |
|---|---|
| p1 | http://BTF2Test1.Schema1 |
| p2 | http://msctsc.sap.corp/xi2biz |

**Figure 16 SAP XI - Expression Editor**

## Configuration I: Routing XI Request message to BizTalk

In order to route the request message to BizTalk the following receiver determination is used (including a SOAP Receiver adapter in the receiver agreement):


Configuration: Integration Builder (p136796_XIP_00)

Display Receiver Determination

Status: Active

**Sender**

| | |
|---|---|
| Party | |
| Service | BPM_XI_BTF2_Integration |
| Interface | BTF2_Request_Root_Abstract |
| Namespace | http://msctsc.sap.corp/xi2biz |

**Receiver**

| | |
|---|---|
| Party | * |
| Service | * |
| Description | |

**Configured Receivers**

| Condition | Party | Service |
|---|---|---|
| | | MSCTSC_BizTalk_2004 |

**Configuration Overview for Receiver Determination**

| Receiver (Partner \| Service) | Mapping | Receiver Agreement (Communication Channel) |
|---|---|---|
| \| MSCTSC_BizTalk_2004 | | |
| BTF2_Request_Root_Inbound | Not Specified | SOAP_RECEIVER_BTF2 |

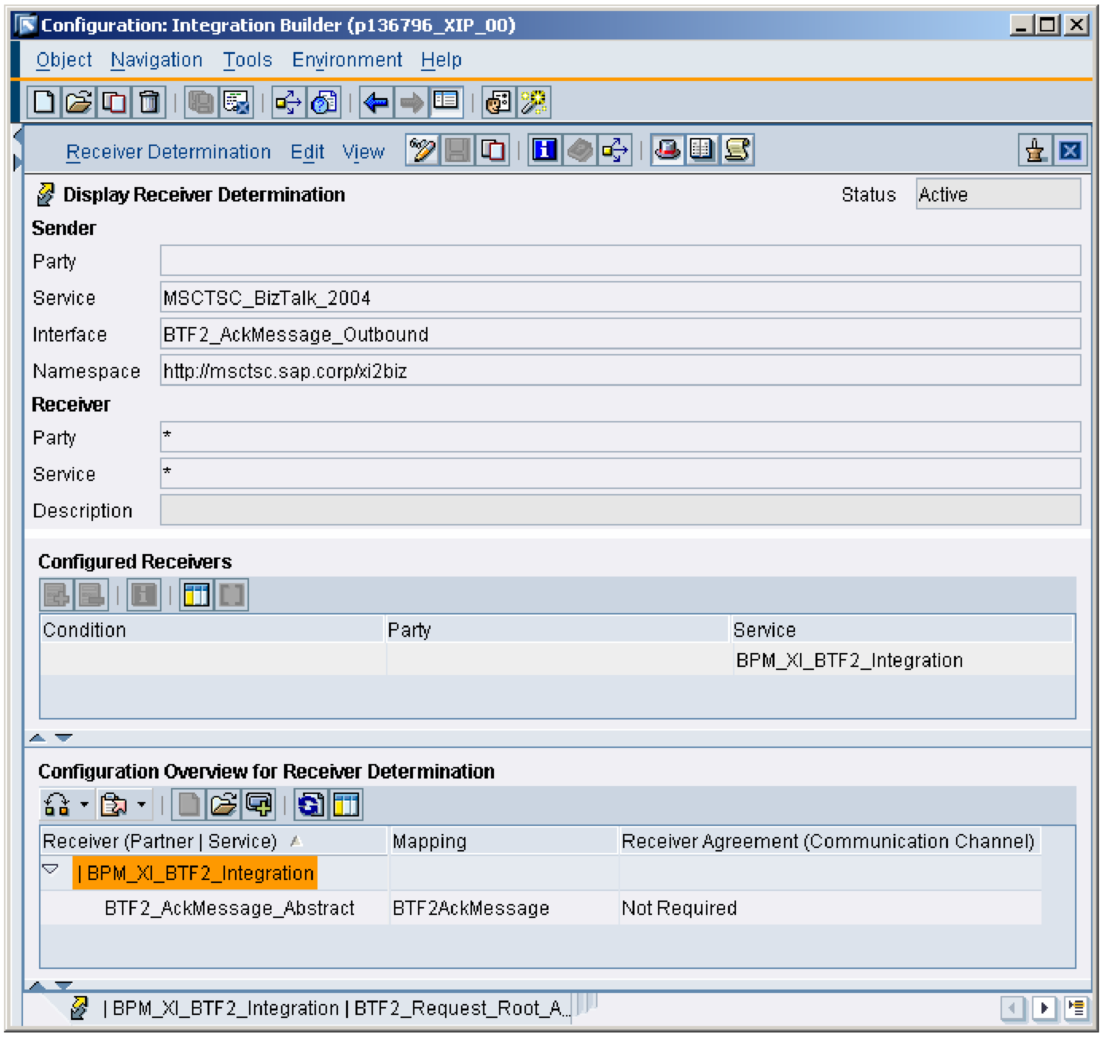
| BPM_XI_BTF2_Integration | BTF2_Request_Root_Abstract

**Figure 17 SAP XI - Receiver Determination**

Note: The SOAP Adapter used in the above receiver agreement is responsible to convert the XI message into a BTF2 compatible message using an own developed adapter module (described in the following of this collaboration brief).

## Configuration II: Routing BTF2 Acknowledgment to ccBPM

The BTF2 acknowledgment is routed back to the BPM process using the receiver determination shown in the next screenshot. In the referenced interface determination an own developed Java Mapping is used to convert the BizTalk BTF2 response into a simple XI message that is delivered to the appropriate BPM process instance using the specified correlation (message contains only one “BTF2_Meta_Information” tag with the appropriate UUID) .


Configuration: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Receiver Determination Edit View

Display Receiver Determination

Status: Active

**Sender**

| | |
|---|---|
| Party | |
| Service | MSCTSC_BizTalk_2004 |
| Interface | BTF2_AckMessage_Outbound |
| Namespace | http://msctsc.sap.corp/xi2biz |

**Receiver**

| | |
|---|---|
| Party | * |
| Service | * |
| Description | |

Configured Receivers

| Condition | Party | Service |
|---|---|---|
| | | BPM_XI_BTF2_Integration |

Configuration Overview for Receiver Determination

| Receiver (Partner \| Service) | Mapping | Receiver Agreement (Communication Channel) |
|---|---|---|
| \| BPM_XI_BTF2_Integration | | |
| BTF2_AckMessage_Abstract | BTF2AckMessage | Not Required |

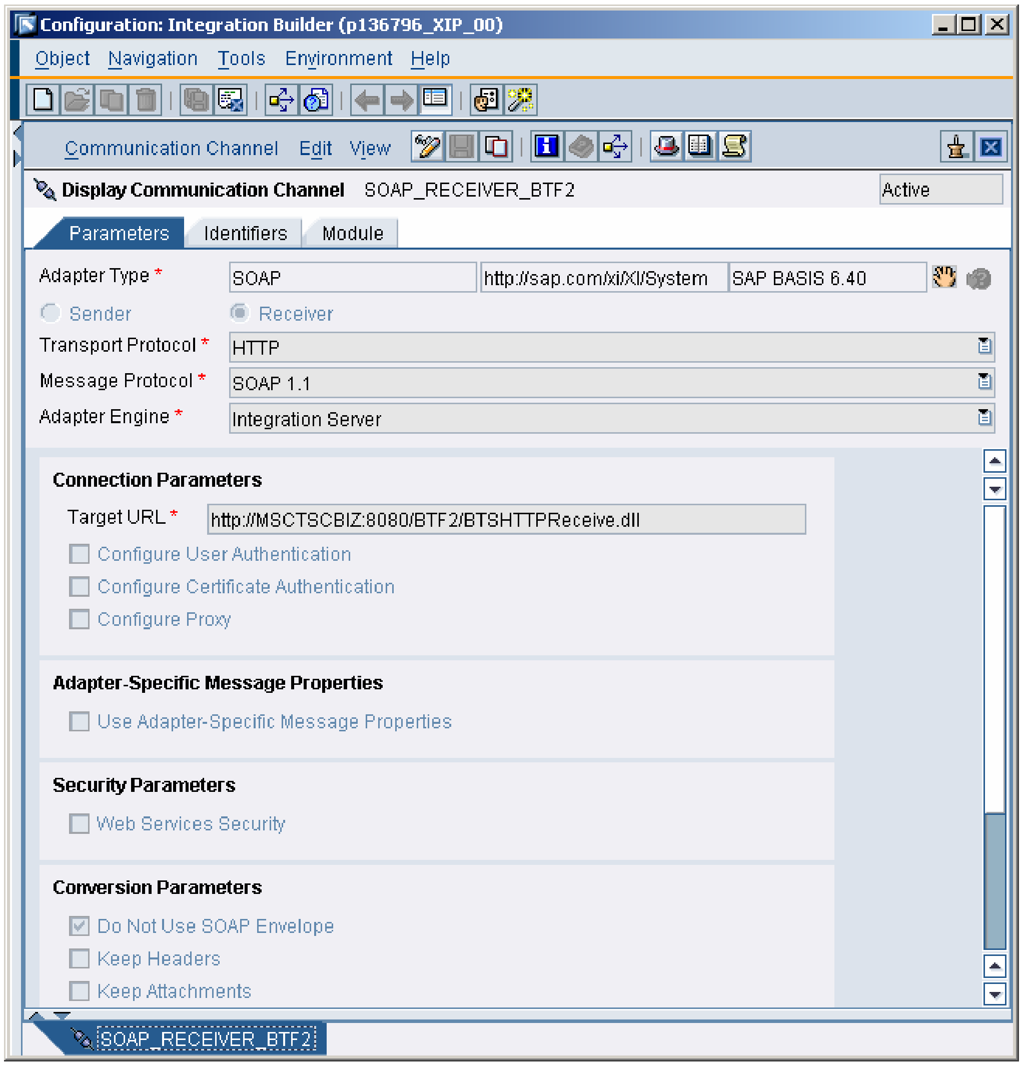
| BPM_XI_BTF2_Integration | BTF2_Request_Root_A...

**Figure 18  SAP XI - Receiver Determination**

## Configuration III: SOAP-Receiver Adapter & BTF2Wrapper Adapter Module

The BTF2 message is created in an own developed Java adapter module that is plugged in to the SOAP-Receiver adapter. An XI Adapter Framework module is defined as a stateless session EJB that implements the Adapter Framework Module interface.

Note: The complete BTF2 SOAP message is created in the adapter module (including the SOAP envelope). Therefore the flag “Do not use SOAP Envelope has to be activated” for the SOAP communication channel.


Configuration: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Communication Channel Edit View
Parameters Identifiers Module

Display Communication Channel SOAP_RECEIVER_BTF2 Active

Adapter Type * SOAP http://sap.com/xi/XI/System SAP BASIS 6.40

Sender [✓ Receiver]

Transport Protocol * HTTP

Message Protocol * SOAP 1.1

Adapter Engine * Integration Server

**Connection Parameters**

Target URL * http://MSCTSCBIZ:8080/BTF2/BTSHTTPReceive.dll

- ☐ Configure User Authentication
- ☐ Configure Certificate Authentication
- ☐ Configure Proxy

**Adapter-Specific Message Properties**

- ☐ Use Adapter-Specific Message Properties

**Security Parameters**

- ☐ Web Services Security

**Conversion Parameters**

- ☑ Do Not Use SOAP Envelope
- ☐ Keep Headers
- ☐ Keep Attachments

SOAP_RECEIVER_BTF2

**Figure 19 SAP XI – SOAP Adapter Configuration**

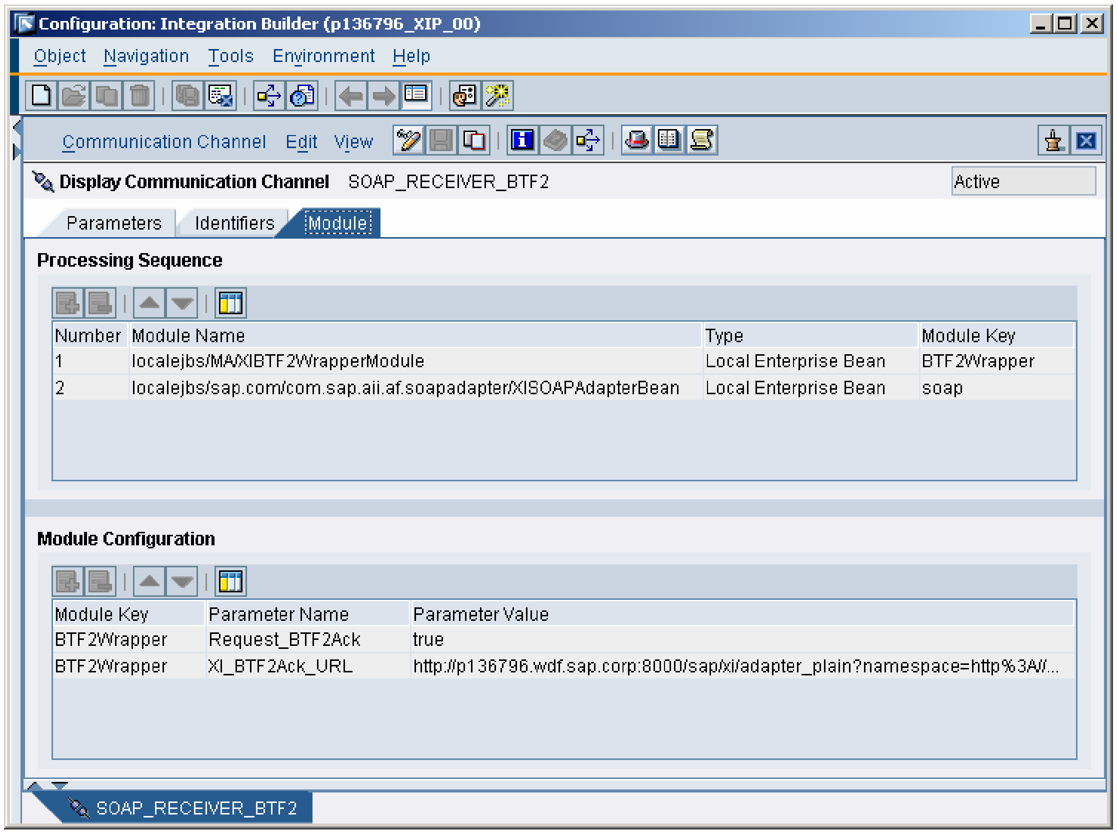
The module reads the message sent by the Integration Server (XI IS Outbound) and creates the BTF2 message. In order to use the Adapter Module it has to be deployed as a SDA file on the J2EE Engine using the SDM (Software Deployment Manager).

In the next table the adapter module configuration parameters are described:

| Parameter Name | Description |
|---|---|
| Request_BTF2Ack | If this flag is set to “true” the BTF2 message created by the adapter module will trigger BizTalk to send back a BTF2 acknowledgement. In order to setup XI/BizTalk integration without a resend mechanism in XI, the request of a BTF2 acknowledgment from BizTalk can be switched off by setting the flag to “false” (Use case scenario 2). |
| XI_BTF2_Ack_URL | HTTP target URL used by BizTalk as the destination of the BTF2 acknowledgment message (e.g. address of XI Plain HTTP-Adapter) |

**Figure 20 SOAP Adapter Module: Configuration Parameters**

Next figure shows the configuration of the BTF2Wrapper module:



**Display Communication Channel** SOAP_RECEIVER_BTF2 Active

**Processing Sequence**

| Number | Module Name | Type | Module Key |
|---|---|---|---|
| 1 | localejbs/MA/XIBTF2WrapperModule | Local Enterprise Bean | BTF2Wrapper |
| 2 | localejbs/sap.com/com.sap.aii.af.soapadapter/XISOAPAdapterBean | Local Enterprise Bean | soap |

**Module Configuration**

| Module Key | Parameter Name | Parameter Value |
|---|---|---|
| BTF2Wrapper | Request_BTF2Ack | true |
| BTF2Wrapper | XI_BTF2Ack_URL | http://p136796.wdf.sap.corp:8000/sap/xi/adapter_plain?namespace=http%3A//... |

**Figure 21 SOAP Adapter Module: Configuration Parameters**

In the next section an extract of the adapter module source code is shown. The BTF2 message is created in the BTF2MessageFactoryP2 class using XML DOM (Document Object Model) API:

```
/**
 * The main method of AF modules is the <code>process()</code> method. It takes
 * the XI message, changes it according to some module specific rules and
 * forwards it in the module chain. If this module is the last module in the
 * chain before the adapter is being called it must ensure that in case of
 * synchronous messages a response message is sent back in the return
 * <code>ModuleDate</code> parameter.
 */
public ModuleData process(
       ModuleContext moduleContext,
       ModuleData inputModuleData)
       throws ModuleException {

       …

       // Read module properties
       requestBTF2Ack = (String)
       moduleContext.getContextData("Request_BTF2Ack");
       btf2AckURL = (String) moduleContext.getContextData("XI_BTF2Ack_URL");

       // Read payload
       obj = inputModuleData.getPrincipalData();
       msgIn = (Message) obj;
       xmlpayloadIn = msgIn.getDocument();
       inputStream = xmlpayloadIn.getInputStream();

       // Construct BTF2 message
       BTF2MessageFactoryP2 messageFactory = new BTF2MessageFactoryP2();
       Document targetDocument =
              messageFactory.createBTF2Message(
                     inputStream,
                     requestBTF2Ack,
                     btf2AckURL);

       …
}
```

**Figure 22 SOAP Adapter Module: Code Extract**

In the next parts of this collaboration brief we will focus in detail on the SOAP messages exchanged by XI and BizTalk (request message from XI to BizTalk and BTF2 acknowledgment form BizTalk to XI):

## Runtime I: BTF2 Request message from XI to BizTalk

The following message is send from the central XI pipeline to the SOAP-Receiver adapter. The actual sender of this message is the BPM process described in the previous parts of this collaboration brief. Note: Message contains an UUID as an attribute of the “BTF2_Meta_Information” tag.

```
<?xml version="1.0" encoding="UTF-8" ?>
 <ns0:Root xmlns:ns0="http://BTF2Test1.Schema1">
       <BTF2_Meta_Information
        UUID="8D002C50A3413A48B979CFADF15CDBB2" />
       <test>SDN Article</test>
</ns0:Root>
```

Before the message is sent to BizTalk the Adapter Framework module processor calls the BTF2WrapperModule outlined in the previous chapter. The adapter module reads the inbound message and transforms it into a BTF2 compliant SOAP message.
The UUID of the "BTF2_Meta_Information" tag is assigned to the "prop:identity" of the BTF2 SOAP message.
Note: If the BTF2WrapperModule is used without a resend mechanism in XI (configuration parameter Request_BTF2Ack is set to "false") the XI MessageID is used and assigned to the "prop:identity" tag of the BTF2 message. In this case the inbound message does not have to implement a "BTF2_Meta_Information" tag.

```
<?xml version="1.0" encoding="UTF8" ?>
 <SOAPENV:Envelope xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchemainstance">
        <SOAPENV:Header>
            <eps:endpoints
                   xmlns:eps="http://schemas.biztalk.org/btf20/endpoints"
                   SOAPENV:mustUnderstand="1"
                   xmlns:biz="http://schemas.biztalk.org/btf20/address/types">
            <eps:to>
              <eps:address xsi:type="biz:OrganizationName">Receiver</eps:address>
            </eps:to>
           <eps:from>
             <eps:address xsi:type="biz:OrganizationName">Sender</eps:address>
          </eps:from>
        </eps:endpoints>
         <prop:properties xmlns:prop="http://schemas.biztalk.org/btf20/properties"
                    SOAPENV:mustUnderstand="1">
          <prop:identity>uuid:8D002C50A3413A48B979CFADF15CDBB2</prop:identity>
          <prop:sentAt>20051020T11:04:21+00:00</prop:sentAt>
          <prop:expiresAt>20051231T15:13:43+00:00</prop:expiresAt>
          <prop:topic>root:Root</prop:topic>
        </prop:properties>
        <services:services
              xmlns:services="http://schemas.biztalk.org/btf20/services">
            <services:deliveryReceiptRequest>
              <services:sendTo>
                <services:address
                  xsi:type="biz:httpURL">http://p136796.wdf.sap.corp:8000/sap/xi/
                  adapter_plain?namespace=http%3A//msctsc.sap.corp/xi2biz&
                  interface=BTF2_AckMessage_Outbound&service=MSCTSC_BizTalk_
                  2004&party=&agency=&scheme=&QOS=EO&sapuser=xiappluser
                  &sappassword=passpass&sapclient=100
              </services:address>
             </services:sendTo>
            <services:sendBy>20051231T15:13:43+00:00</services:sendBy>
          </services:deliveryReceiptRequest>
        </services:services>
</SOAPENV:Header>
<SOAPENV:Body>
        <ns0:Root xmlns:ns0="http://BTF2Test1.Schema1">
                <test>SDN Article</test>
        </ns0:Root>
</SOAPENV:Body>
</SOAPENV:Envelope>
```

**Figure 23 BTF2 Message Send from XI to BizTalk**

After sending the BTF2 SOAP message BizTalk responds with the following HTTP message.

```
HTTP/1.1 202 Message Accepted
Connection: close
Date: Thu, 20 Oct 2005 08:57:41 GMT
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
Content-length: 136
Content-Type: text/xml; charset=utf-8

<BizTalkHttpReceive>
   <CorrelationToken TokenType="SUBMIT">{0A1F2389-01CE-46DE-BE98-
73F9D5F68B26}</CorrelationToken></BizTalkHttpReceive>
```

**Figure 24 HTTP Response BizTalk**

Note: This is only the response to the BTF2 SOAP request and not the BTF2 acknowledgment itself.

## Runtime II: BTF2 Acknowledgement Message from BizTalk to XI

After receiving the BTF2 compliant SOAP message BizTalk sends back a BTF acknowledgment to the HTTP address specified in the "services:address" tag of the BTF2 request message.

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAPENV:Envelope xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
      xmlns:biz="http://schemas.biztalk.org/btf20/address/types">
  <SOAPENV:Header>
      <eps:endpoints SOAPENV:mustUnderstand="1"
      xmlns:eps="http://schemas.biztalk.org/btf20/endpoints"
      xmlns:biz="http://schemas.biztalk.org/btf20/address/types">
        <eps:to>
             <eps:address xsi:type="biz:httpURL" >
             http://p136796.wdf.sap.corp:8000/sap/xi/adapter_plain?namespace
             =http%3A//msctsc.sap.corp/xi2biz&interface=BTF2_AckMessage
             _Outbound&service=MSCTSC_BizTalk_2004&party=&agency=
             &scheme=&QOS=EO&sapuser=xiappluser&sappassword
             =passpass&sapclient=100
             </eps:address>
        </eps:to>
        <eps:from>
             <eps:address xsi:type="biz:OrganizationName">Receiver</eps:address>
       </eps:from>
      </eps:endpoints>
      <prop:properties SOAPENV:mustUnderstand="1"
            xmlns:prop="http://schemas.biztalk.org/btf20/properties">
        <prop:identity>uuid:b7b703c6b114475f8b8bfd732c37c437</prop:identity>
             <prop:sentAt>20051020T08:57:41+00:00</prop:sentAt>
             <prop:expiresAt>20051231T15:13:43+00:00</prop:expiresAt>
        <prop:topic>http://schemas.biztalk.org/btf20/receipts/deliveryReceipt
        </prop:topic>
      </prop:properties>
```

```
        <rcpt:deliveryReceipt SOAPENV:mustUnderstand="1"
           xmlns:rcpt="http://schemas.biztalk.org/btf20/receipts">
          <rcpt:receivedAt>20051020T08:57:41+00:00</rcpt:receivedAt>
          <rcpt:identity>uuid:8D002C50A3413A48B979CFADF15CDBB2</rcpt:identity>
        </rcpt:deliveryReceipt>
        </SOAPENV:Header>
        <SOAPENV:Body />
</SOAPENV:Envelope>
```

**Figure 25 BTF2 Acknowledgement Message**

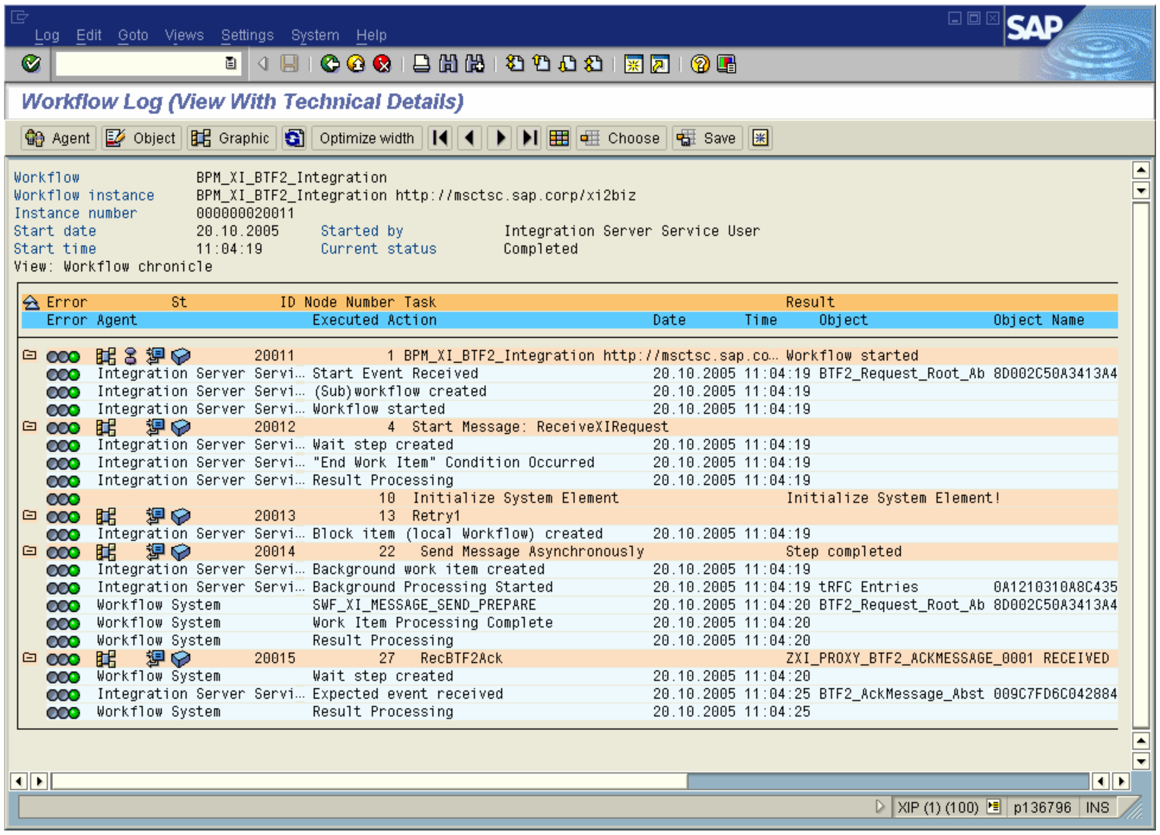
After receiving the BTF2 acknowledgment the Plain HTTP-Adapter of XI response with the following HTTP response code:

```
HTTP/1.1 100 Continue
Server: SAP Web Application Server
HTTP/1.1 200 Message accepted
set-cookie: sap-usercontext=sap-client=100; path=/
content-type: text/html
content-length: 0
msgguid: 009C7FD6C042884CBA87E5B9DEF38CD5
server: SAP Web Application Server (1.0;640)
```

**Figure 26 HTTP Response from XI**

## Runtime III: BPM Process in XI

The UUID contained in the tag “rcpt:identity” of the BTF2 acknowledgment message matches the “prop:identity” tag value of the BTF2 request message. As described in the previous sections this criteria is used in a XI BPM correlation definition to match the request and response message. In the next screenshot you can see the technical workflow protocol of a completed BPM_XI_BTF2_Integration process in XI:

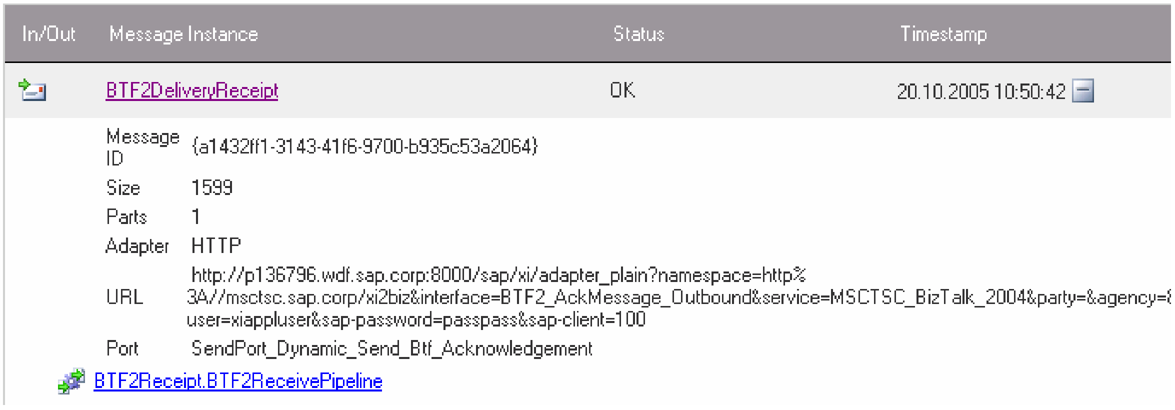

Log Edit Goto Views Settings System Help
Workflow Log (View With Technical Details)
Agent Object Graphic Optimize width Choose Save

```
Workflow             BPM_XI_BTF2_Integration
Workflow instance    BPM_XI_BTF2_Integration http://msctsc.sap.corp/xi2biz
Instance number      000000020011
Start date           20.10.2005     Started by        Integration Server Service User
Start time           11:04:19       Current status    Completed
View: Workflow chronicle

Error          St            ID Node Number Task                                  Result
Error Agent                  Executed Action                          Date       Time     Object                Object Name

                             20011        1 BPM_XI_BTF2_Integration http://msctsc.sap.co… Workflow started
Integration Server Servi…    Start Event Received                     20.10.2005 11:04:19 BTF2_Request_Root_Ab 8D002C50A3413A4
Integration Server Servi…    (Sub)workflow created                    20.10.2005 11:04:19
Integration Server Servi…    Workflow started                         20.10.2005 11:04:19
                             20012        4  Start Message: ReceiveXIRequest
Integration Server Servi…    Wait step created                        20.10.2005 11:04:19
Integration Server Servi…    "End Work Item" Condition Occurred       20.10.2005 11:04:19
Integration Server Servi…    Result Processing                        20.10.2005 11:04:19
                                         10  Initialize System Element                    Initialize System Element!
                             20013       13  Retry1
Integration Server Servi…    Block item (local Workflow) created      20.10.2005 11:04:19
                             20014       22   Send Message Asynchronously                 Step completed
Integration Server Servi…    Background work item created             20.10.2005 11:04:19
Integration Server Servi…    Background Processing Started            20.10.2005 11:04:19 tRFC Entries         0A1210310A8C435
Workflow System              SWF_XI_MESSAGE_SEND_PREPARE              20.10.2005 11:04:20 BTF2_Request_Root_Ab 8D002C50A3413A4
Workflow System              Work Item Processing Complete            20.10.2005 11:04:20
Workflow System              Result Processing                        20.10.2005 11:04:20
                             20015       27   RecBTF2Ack                                  ZXI_PROXY_BTF2_ACKMESSAGE_0001 RECEIVED
Workflow System              Wait step created                        20.10.2005 11:04:20
Integration Server Servi…    Expected event received                  20.10.2005 11:04:25 BTF2_AckMessage_Abst 009C7FD6C042884
Workflow System              Result Processing                        20.10.2005 11:04:25
```

**Figure 27 SAP XI: Workflow Log**

## Runtime IV: Message in BizTalk

Within the BizTalk Monitoring Tool HAT (Health and Activity Tracking) you can see the triggering of the acknowledgment message within the BTF2ReceivePipeline.



| In/Out | Message Instance | Status | Timestamp |
|---|---|---|---|
|  | BTF2DeliveryReceipt | OK | 20.10.2005 10:50:42 |

| | |
|---|---|
| Message ID | {a1432ff1-3143-41f6-9700-b935c53a2064} |
| Size | 1599 |
| Parts | 1 |
| Adapter | HTTP |
| URL | http://p136796.wdf.sap.corp:8000/sap/xi/adapter_plain?namespace=http%3A//msctsc.sap.corp/xi2biz&interface=BTF2_AckMessage_Outbound&service=MSCTSC_BizTalk_2004&party=&agency=&user=xiappluser&sap-password=passpass&sap-client=100 |
| Port | SendPort_Dynamic_Send_Btf_Acknowledgement |

BTF2Receipt.BTF2ReceivePipeline

**Figure 28 BizTalk: Monitoring**

## Setup of the BizTalk Part

### Overview

On the BizTalk side we configured a simple orchestration that receives messages from SAP XI using the HTTP adapter. The Receive Port *BTF2inPort* is configured to use BizTalk Framework 2.0 to achieve reliable messaging. The payload of the message is then send to the local file system of the BizTalk Server using the Port *MessageOut*.

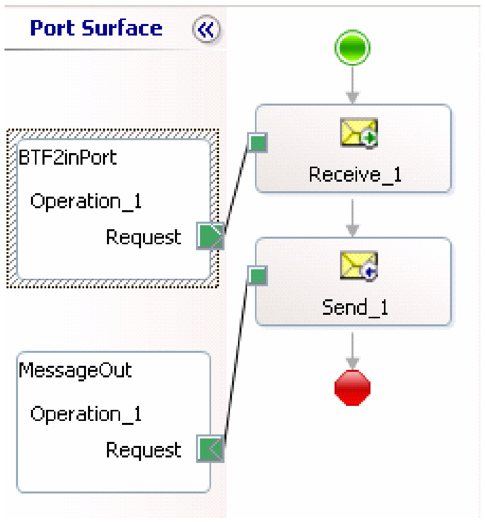
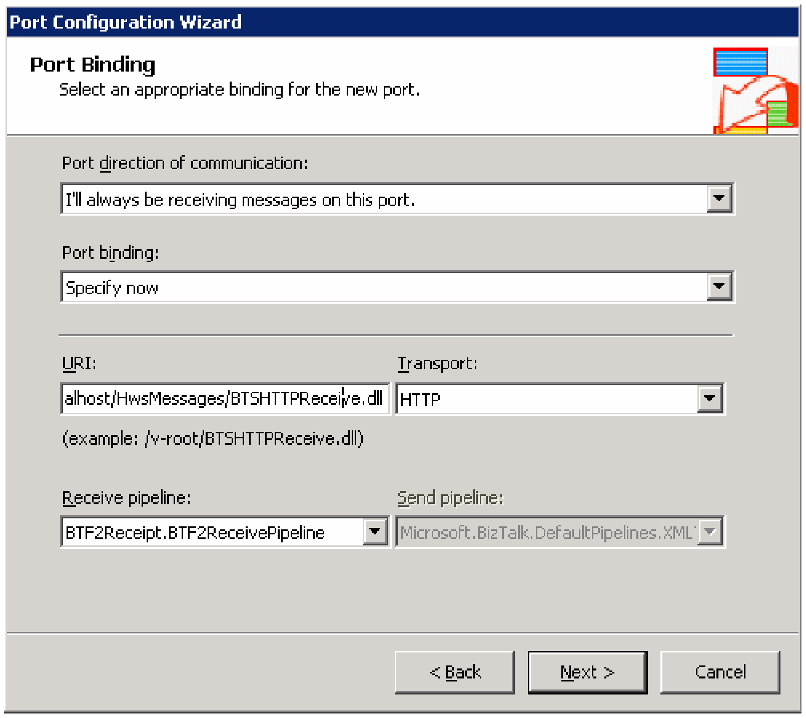


**Figure 29  Orchestration Used to Receive BTF2 Compliant Documents from SAP XI**

### Receive Port

The BTF2 document is received by the port BTF2inPort that is configured to use the Receive Pipeline *BTF2Receipt.BTF2ReceivePipeline*.



Port Binding

Select an appropriate binding for the new port.

Port direction of communication: I'll always be receiving messages on this port.

Port binding: Specify now

URI: alhost/HwsMessages/BTSHTTPReceive.dll

Transport: HTTP

(example: /v-root/BTSHTTPReceive.dll)

Receive pipeline: BTF2Receipt.BTF2ReceivePipeline

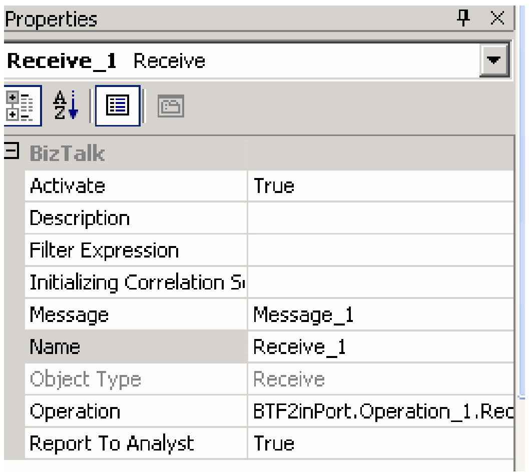
Send pipeline: Microsoft.BizTalk.DefaultPipelines.XML

**Figure 30 BizTalk: Receive Port**

The BTF2.Receivepipeline performs the duplicate check and will send (if requested to do so) the acknowledgement to the address specified in the tag delivery receipt request in the BizTalk Framework document.

## Orchestration

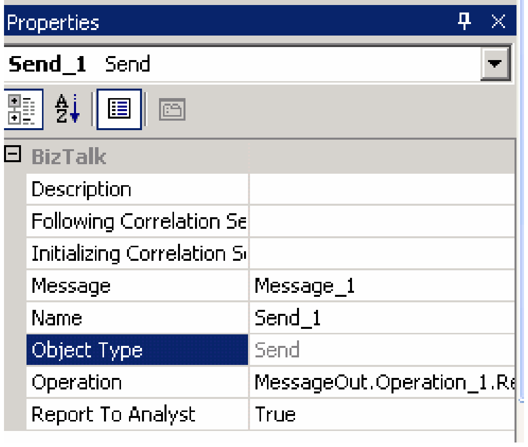
The orchestration only consists out of a receive shape that receives messages of type Message1 and a send shape that sends those messages to the file system.



**Receive_1** Receive

BizTalk

| | |
|---|---|
| Activate | True |
| Description | |
| Filter Expression | |
| Initializing Correlation S | |
| Message | Message_1 |
| Name | Receive_1 |
| Object Type | Receive |
| Operation | BTF2inPort.Operation_1.Rec |
| Report To Analyst | True |


Properties

Send_1 Send

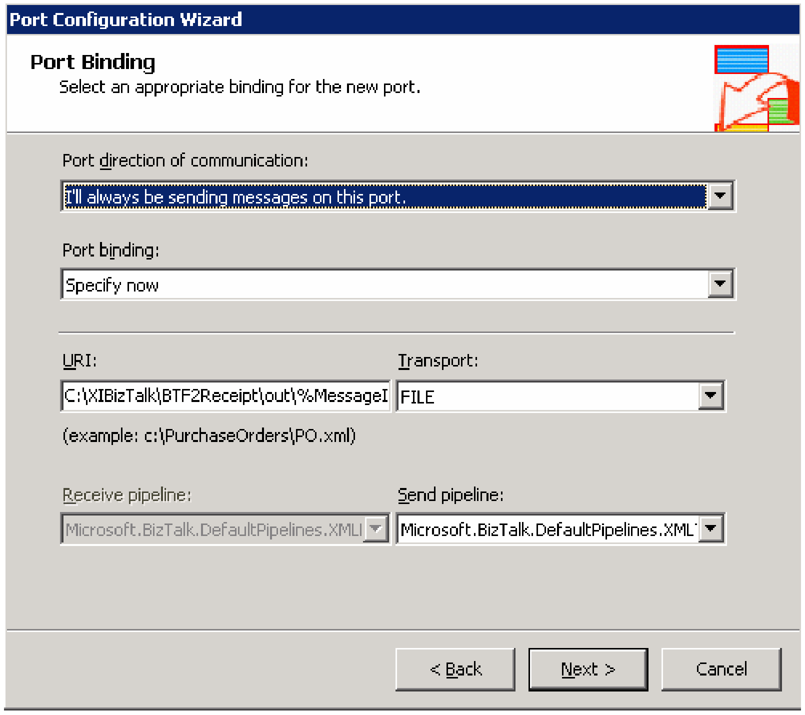
BizTalk

| | |
|---|---|
| Description | |
| Following Correlation Se | |
| Initializing Correlation S | |
| Message | Message_1 |
| Name | Send_1 |
| Object Type | Send |
| Operation | MessageOut.Operation_1.Re |
| Report To Analyst | True |

**Figure 31 BizTalk: Orchestration**

Send Port

The payload of the XML message received from SAP XI is send to the local file system (C:\XIBizTalk\BTF2Receipt\out\%MessageID&.xml)


Port Configuration Wizard

Port Binding

Select an appropriate binding for the new port.

Port direction of communication: I'll always be sending messages on this port.

Port binding: Specify now

URI: C:\XIBizTalk\BTF2Receipt\out\%MessageI

Transport: FILE

(example: c:\PurchaseOrders\PO.xml)

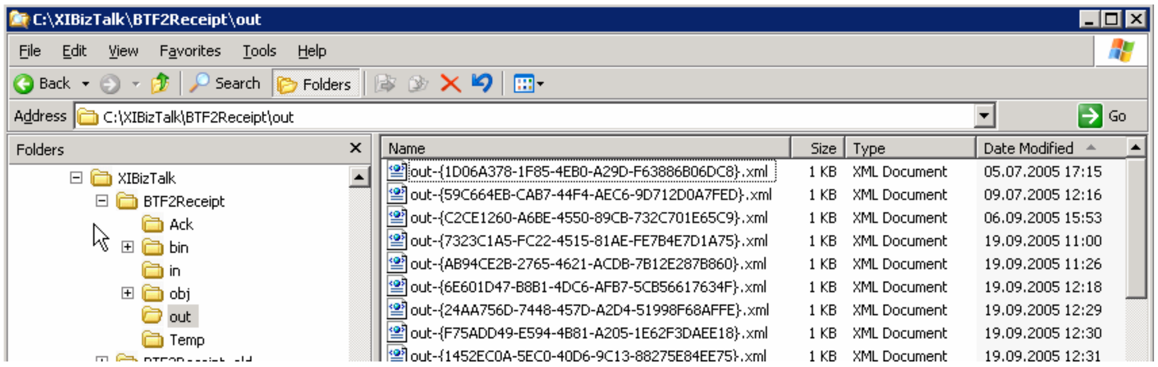
Receive pipeline: Microsoft.BizTalk.DefaultPipelines.XMLI

Send pipeline: Microsoft.BizTalk.DefaultPipelines.XML1

**Figure 32 BizTalk: Send Port**

## Result

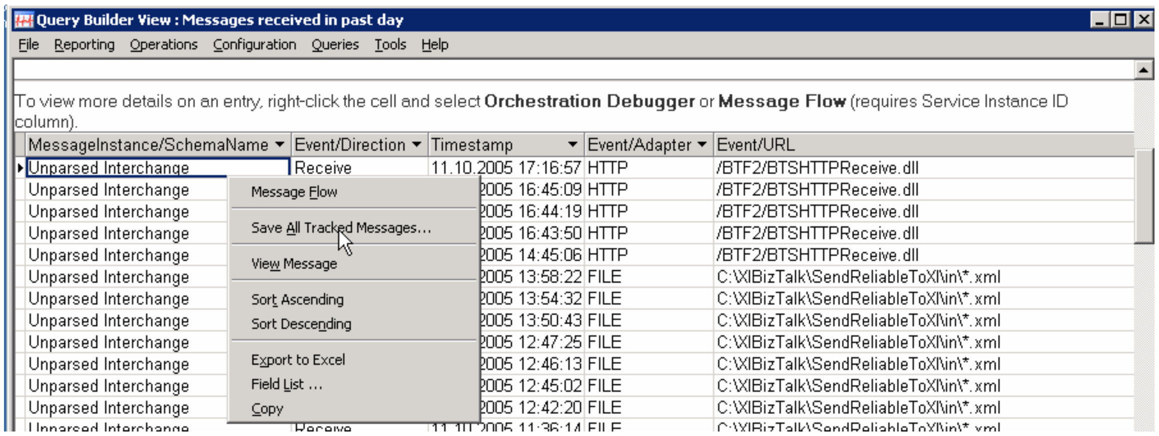
The messages that have been successfully sent reliable from SAP XI to Microsoft BizTalk are finally stored on the local file system.


C:\XIBizTalk\BTF2Receipt\out
File Edit View Favorites Tools Help
Back Search Folders
Address C:\XIBizTalk\BTF2Receipt\out
Go
Folders
XIBizTalk
BTF2Receipt
Ack
bin
in
obj
out
Temp

| Name | Size | Type | Date Modified |
|---|---|---|---|
| out-{1D06A378-1F85-4EB0-A29D-F63886B06DC8}.xml | 1 KB | XML Document | 05.07.2005 17:15 |
| out-{59C664EB-CAB7-44F4-AEC6-9D712D0A7FED}.xml | 1 KB | XML Document | 09.07.2005 12:16 |
| out-{C2CE1260-A6BE-4550-89CB-732C701E65C9}.xml | 1 KB | XML Document | 06.09.2005 15:53 |
| out-{7323C1A5-FC22-4515-81AE-FE7B4E7D1A75}.xml | 1 KB | XML Document | 19.09.2005 11:00 |
| out-{AB94CE2B-2765-4621-ACDB-7B12E287B860}.xml | 1 KB | XML Document | 19.09.2005 11:26 |
| out-{6E601D47-B8B1-4DC6-AFB7-5CB56617634F}.xml | 1 KB | XML Document | 19.09.2005 12:18 |
| out-{24AA756D-7448-457D-A2D4-51998F68AFFE}.xml | 1 KB | XML Document | 19.09.2005 12:29 |
| out-{F75ADD49-E594-4B81-A205-1E62F3DAEE18}.xml | 1 KB | XML Document | 19.09.2005 12:30 |
| out-{1452EC0A-5EC0-40D6-9C13-88275E84EE75}.xml | 1 KB | XML Document | 19.09.2005 12:31 |

**Figure 33  BizTalk: XML Messages Send to File System**

Using the BizTalk Monitoring Tool HAT (Health and Activity Tracking) it is possible to check for the flow of a message inside BizTalk.

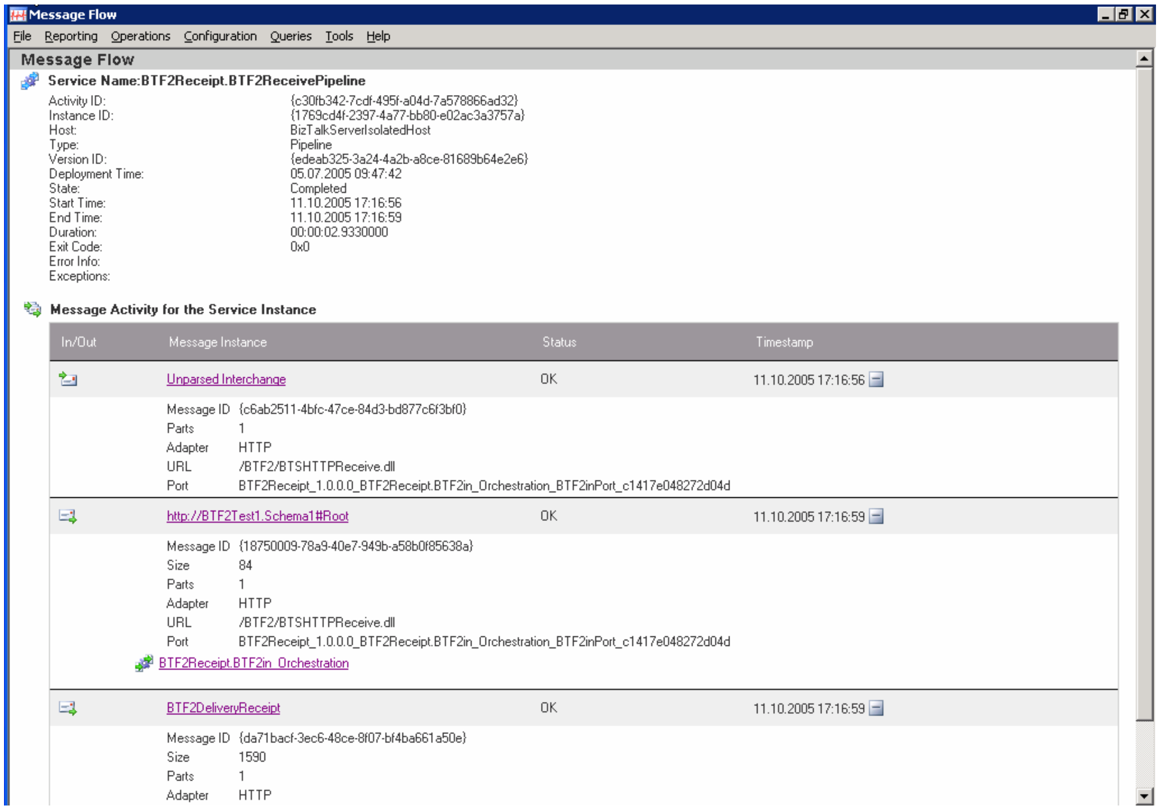

Query Builder View : Messages received in past day
File Reporting Operations Configuration Queries Tools Help
Message Flow
Save All Tracked Messages…
View Message
Sort Ascending
Sort Descending
Export to Excel
Field List …
Copy

To view more details on an entry, right-click the cell and select **Orchestration Debugger** or **Message Flow** (requires Service Instance ID column).

| MessageInstance/SchemaName | Event/Direction | Timestamp | Event/Adapter | Event/URL |
|---|---|---|---|---|
| Unparsed Interchange | Receive | 11.10.2005 17:16:57 | HTTP | /BTF2/BTSHTTPReceive.dll |
| Unparsed Interchange | | 2005 16:45:09 | HTTP | /BTF2/BTSHTTPReceive.dll |
| Unparsed Interchange | | 2005 16:44:19 | HTTP | /BTF2/BTSHTTPReceive.dll |
| Unparsed Interchange | | 2005 16:43:50 | HTTP | /BTF2/BTSHTTPReceive.dll |
| Unparsed Interchange | | 2005 14:45:06 | HTTP | /BTF2/BTSHTTPReceive.dll |
| Unparsed Interchange | | 2005 13:58:22 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 13:54:32 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 13:50:43 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 12:47:25 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 12:46:13 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 12:45:02 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | | 2005 12:42:20 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |
| Unparsed Interchange | Receive | 11.10.2005 11:36:14 | FILE | C:\XIBizTalk\SendReliableToXI\in\*.xml |

**Figure 34  BizTalk: Monitoring**

To view details one has to right click the message instance and select **Message Flow**.



Message Flow

Service Name:BTF2Receipt.BTF2ReceivePipeline

| | |
|---|---|
| Activity ID: | {c30fb342-7cdf-495f-a04d-7a578866ad32} |
| Instance ID: | {1769cd4f-2397-4a77-bb80-e02ac3a3757a} |
| Host: | BizTalkServerIsolatedHost |
| Type: | Pipeline |
| Version ID: | {edeab325-3a24-4a2b-a8ce-81689b64e2e6} |
| Deployment Time: | 05.07.2005 09:47:42 |
| State: | Completed |
| Start Time: | 11.10.2005 17:16:56 |
| End Time: | 11.10.2005 17:16:59 |
| Duration: | 00:00:02.9330000 |
| Exit Code: | 0x0 |
| Error Info: | |
| Exceptions: | |

Message Activity for the Service Instance

| In/Out | Message Instance | Status | Timestamp |
|---|---|---|---|
| | Unparsed Interchange | OK | 11.10.2005 17:16:56 |
| | Message ID {c6ab2511-4bfc-47ce-84d3-bd877c6f3bf0}<br>Parts 1<br>Adapter HTTP<br>URL /BTF2/BTSHTTPReceive.dll<br>Port BTF2Receipt_1.0.0.0_BTF2Receipt.BTF2in_Orchestration_BTF2inPort_c1417e048272d04d | | |
| | http://BTF2Test1.Schema1#Root | OK | 11.10.2005 17:16:59 |
| | Message ID {18750009-78a9-40e7-949b-a58b0f85638a}<br>Size 84<br>Parts 1<br>Adapter HTTP<br>URL /BTF2/BTSHTTPReceive.dll<br>Port BTF2Receipt_1.0.0.0_BTF2Receipt.BTF2in_Orchestration_BTF2inPort_c1417e048272d04d<br>BTF2Receipt.BTF2in_Orchestration | | |
| | BTF2DeliveryReceipt | OK | 11.10.2005 17:16:59 |
| | Message ID {da71bacf-3ec6-48ce-8f07-bf4ba661a50e}<br>Size 1590<br>Parts 1<br>Adapter HTTP | | |

**Figure 35 BizTalk: Message Flow**

## Reliable Messaging from BizTalk to XI

### Overview

In this setup we used the sender SOAP adapter of SAP XI to achieve reliable messaging from BizTalk Server to SAP XI. The SOAP adapter of SAP XI is configured such that the messages are to be processed using the Quality of Service **`Exactly Once`** (asynchronous processing).
Therefore it is necessary that the BizTalk Server sends a SAP XI compliant SOAP message to SAP XI. Since it was easier to implement we chose the approach that BizTalk sends the appropriate values to achieve reliable messaging as parameters in a URL rather than creating a SAP XI specific SOAP header.

As in the scenario reliable messaging from XI to BizTalk there are two use cases.

1. BizTalk/XI integration without a resend mechanism
2. BizTalk/XI integration including a resend mechanism. Based on the http response of the SAP SOAP adapter it would be possible to implement a resend mechanism using the BizTalk orchestration. For this the send port property "Delivery Notification" has to be to "Transmitted". The http response that contains the information whether the SOAP message was send successfully (http 200), whether the duplicate check has thrown a SOAP exception or any other error (for example http 400) can be obtained from the response of the SAP XI SOAP adapter.

Though technically feasible we have not tested the resend mechanism described yet. In the following we will described the first use case.

### Configuration of the BizTalk Side

#### Overview

The orchestration on the BizTalk side consists out of a File adapter that receives XML messages in the local file system (C:\XIBizTalk\). In the orchestration the message that is sent to XI is created. This message is sent to SAP XI using the SOAP adapter. The SOAP adapter in BizTalk is created based on the information stored in the WSDL file that is downloaded from SAP XI and used as a web reference for the BizTalk orchestration. A dynamic port is used to add parameters to the URL that is used to post the SOAP message to SAP XI using the message assignment shape. The orchestration also adds credentials to the message send by BizTalk that are used for Basic authentication.

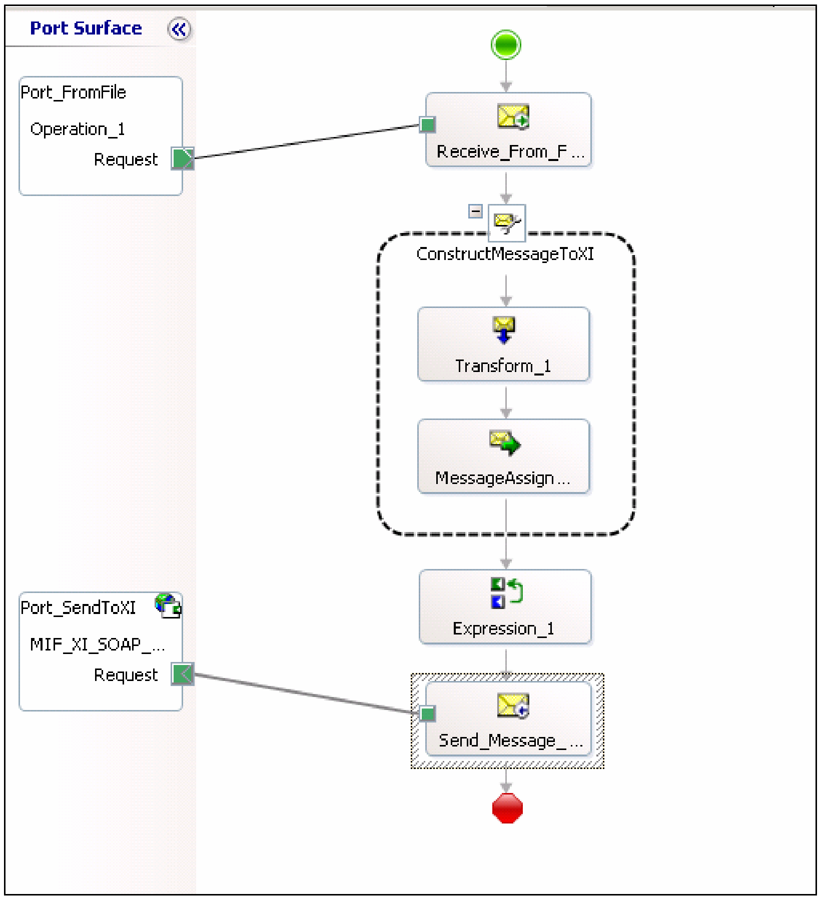


**Figure 36 BizTalk: Orchestration used for Reliable Messaging from BizTalk to SAP XI**

## Receive Port

The orchestration receives XML Messages that are posted in the local file system (e.g. C:\XIBizTalk\SendReliableToXI\In\*.xml). From the message that is received by the *Receive_From_File* Shape a new message is constructed that will be send to the SAP XI SOAP adapter.

## Construct Message shape

In the **Construct Message** shape that we use to construct the Web message we add a Message Assignment shape. In the **Message Assignment** shape we added the following expressions:

```
MessageToXI(SOAP.Username)="xiappluser";
MessageToXI(SOAP.Password)="passpass";
MessageToXI(SOAP.AuthenticationScheme)="Basic";
```

Instead of using a hard coded username and password one could either use Single Sign-On (SSO) that provides mapped user credentials or client certificates.

## Expression Shape

In the **Expression** shape that is located in the orchestration before the message is sent to XI we determine the message ID of the BizTalk message. We have to use the message MessageFromXI rather than the MessageSendToXI since the latter has not been assigned a message ID by BizTalk.

The BizTalk message ID is assigned to a string that is added as an additional parameter to the URL that is called by the dynamic port.

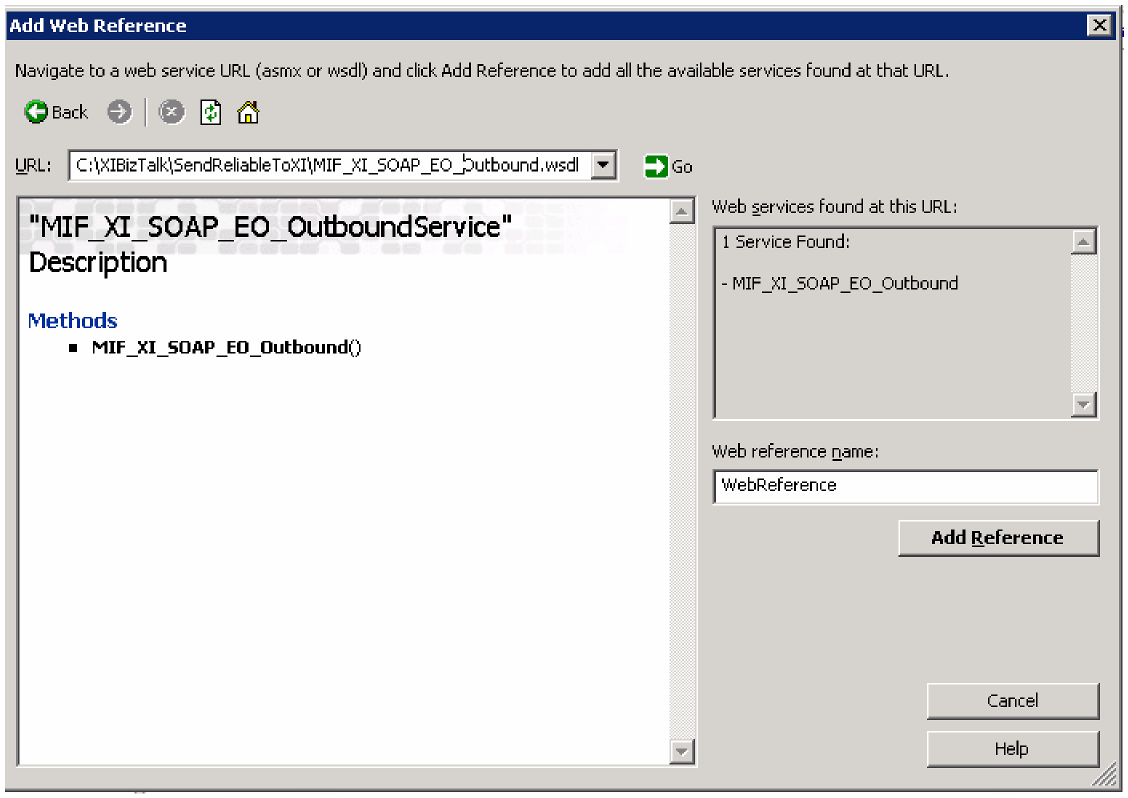
```
myMessageID = MessageFromXI(BTS.MessageID);
Port_SendToXI(Microsoft.XLANGs.BaseTypes.Address)=
"http://p136796:50000/XISOAPAdapter/MessageServlet?channel=:MSCTSC_BizTa
lk_2004:SOAP_Sender_BizTalk&version=3.0&MessageId=" + myMessageID;
```

As a result the following URL will be used by the SOAP adapter of the Microsoft BizTalk Server to call the web service provided by SAP XI.

http://p136796:50000/XISOAPAdapter/MessageServlet?channel=:MSCTSC_BizTalk_2004:SOAP_Sender_BizTalk&MessageId=db29a811-b331-11d9-b3dc-cc3b0a126058&version=3.0

## Send Port

The WSDL File MIF_XI_SOAP_EO_Outbound.wsdl that is obtained from SAP XI is locally stored on the BizTalk Server and added as a Web Reference.


Add Web Reference

Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: C:\XIBizTalk\SendReliableToXI\MIF_XI_SOAP_EO_Outbound.wsdl

"MIF_XI_SOAP_EO_OutboundService" Description

Methods
- MIF_XI_SOAP_EO_Outbound()

Web services found at this URL:

1 Service Found:

- MIF_XI_SOAP_EO_Outbound

Web reference name: WebReference

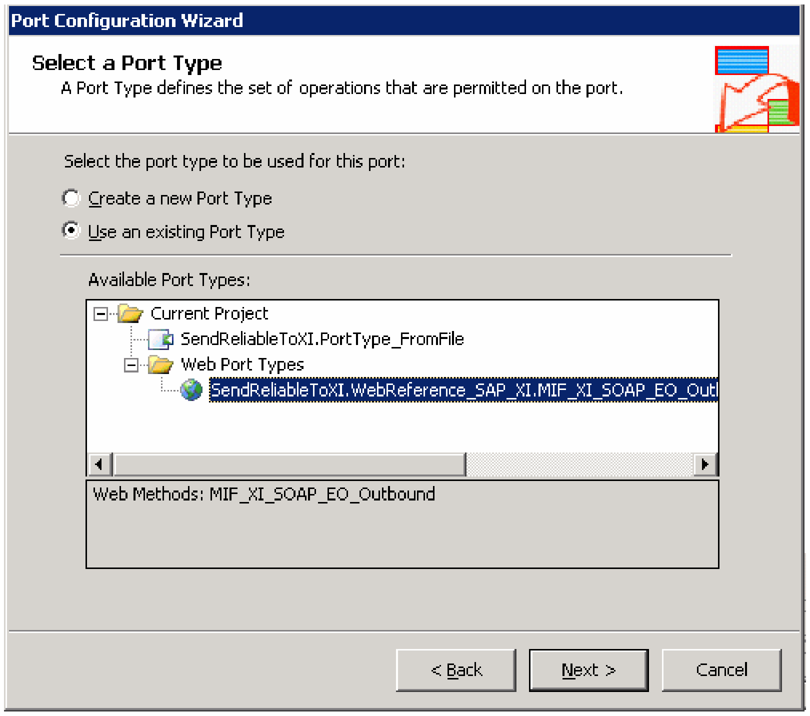
**Figure 37 BizTalk: Create Send Port Using SAP XI WSDL**

As described above it is possible to send messages to SAP XI in a reliable way by passing appropriate URL parameters rather than creating a SAP XI specific SOAP header. The URL that has to be used must have the following format:

```
http://host:port
/XISOAPAdapter/MessageServlet?channel=party:service:channel&
version=3.0&...&MessageId=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

where xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx is a GUID string

In the BizTalk orchestration the SOAP port is configured as a dynamic port which allows setting the URI of the consumed web service provided by SAP XI dynamically. The port is created based on existing port types that are provided as a Web Reference in the previous step.


Port Configuration Wizard

Select a Port Type

A Port Type defines the set of operations that are permitted on the port.

Select the port type to be used for this port:

( ) Create a new Port Type
(●) Use an existing Port Type

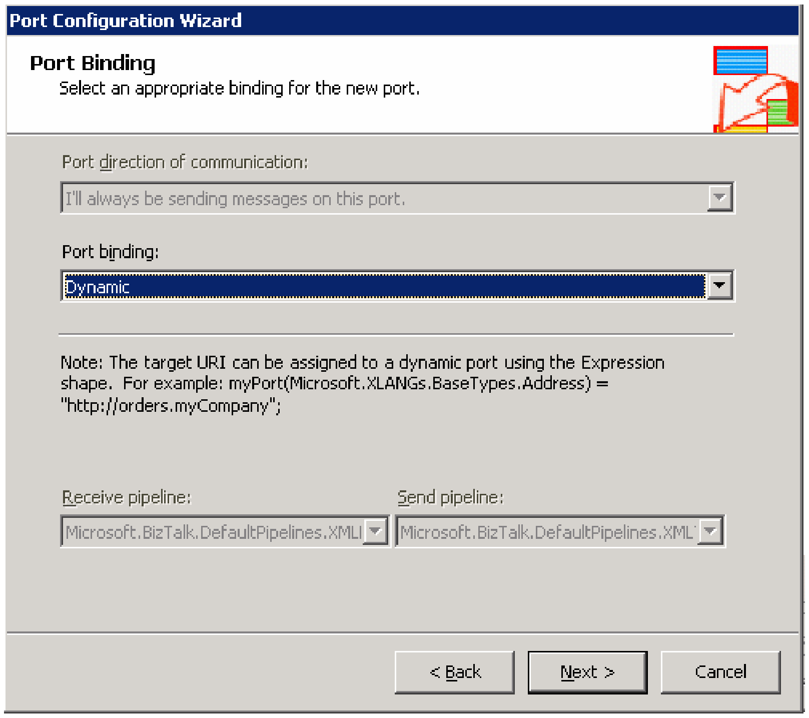
Available Port Types:

- Current Project
  - SendReliableToXI.PortType_FromFile
  - Web Port Types
    - SendReliableToXI.WebReference_SAP_XI.MIF_XI_SOAP_EO_Outb

Web Methods: MIF_XI_SOAP_EO_Outbound

**Figure 38 BizTalk: Create Send Port**

Dynamic port binding is chosen.



Port Configuration Wizard

**Port Binding**

Select an appropriate binding for the new port.

Port direction of communication: I'll always be sending messages on this port.

Port binding: Dynamic

Note: The target URI can be assigned to a dynamic port using the Expression shape. For example: myPort(Microsoft.XLANGs.BaseTypes.Address) = "http://orders.myCompany";

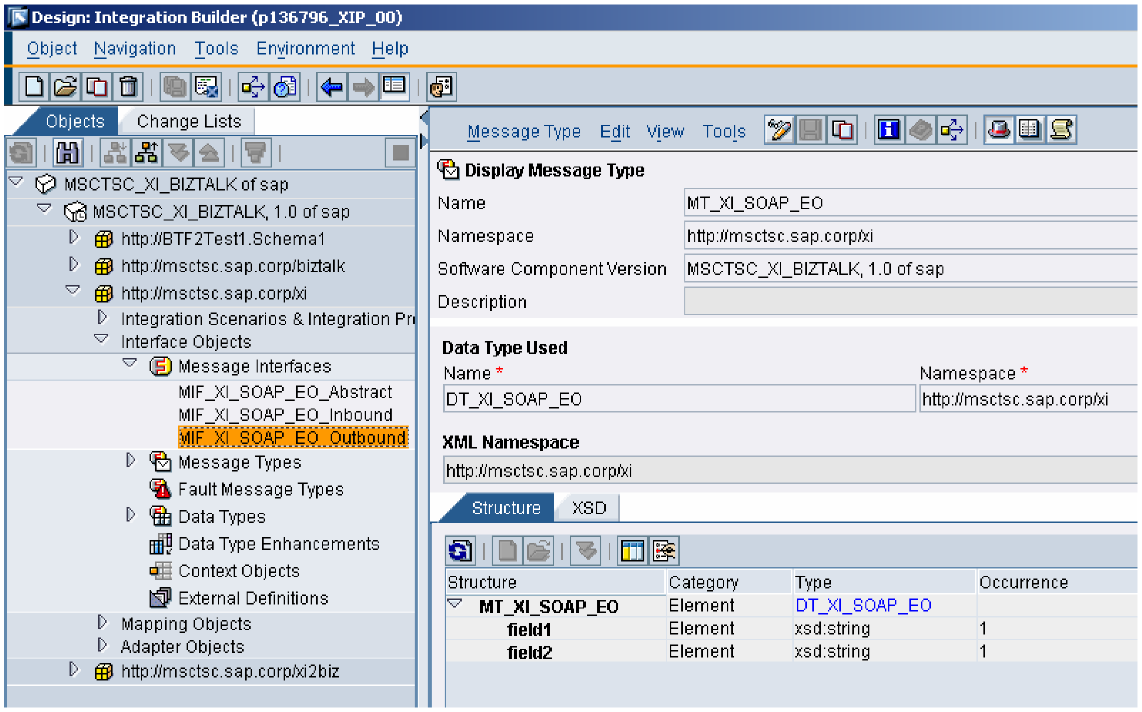
Receive pipeline: Microsoft.BizTalk.DefaultPipelines.XMLI

Send pipeline: Microsoft.BizTalk.DefaultPipelines.XML

**Figure 39 BizTalk: Dynamic Port Configuration**

## Setup of the XI part

In XI an asynchronous message interface is created that will be used in the SOAP-Sender adapter:



Design: Integration Builder (p136796_XIP_00)

Object Navigation Tools Environment Help
Objects Change Lists
Message Type Edit View Tools

```
MSCTSC_XI_BIZTALK of sap
  MSCTSC_XI_BIZTALK, 1.0 of sap
    http://BTF2Test1.Schema1
    http://msctsc.sap.corp/biztalk
    http://msctsc.sap.corp/xi
      Integration Scenarios & Integration Pr
      Interface Objects
        Message Interfaces
          MIF_XI_SOAP_EO_Abstract
          MIF_XI_SOAP_EO_Inbound
          MIF_XI_SOAP_EO_Outbound
        Message Types
        Fault Message Types
        Data Types
        Data Type Enhancements
        Context Objects
        External Definitions
      Mapping Objects
      Adapter Objects
    http://msctsc.sap.corp/xi2biz
```

**Display Message Type**

| | |
|---|---|
| Name | MT_XI_SOAP_EO |
| Namespace | http://msctsc.sap.corp/xi |
| Software Component Version | MSCTSC_XI_BIZTALK, 1.0 of sap |
| Description | |

**Data Type Used**

| Name * | Namespace * |
|---|---|
| DT_XI_SOAP_EO | http://msctsc.sap.corp/xi |

**XML Namespace**

http://msctsc.sap.corp/xi

| Structure | Category | Type | Occurrence |
|---|---|---|---|
| MT_XI_SOAP_EO | Element | DT_XI_SOAP_EO | |
| field1 | Element | xsd:string | 1 |
| field2 | Element | xsd:string | 1 |

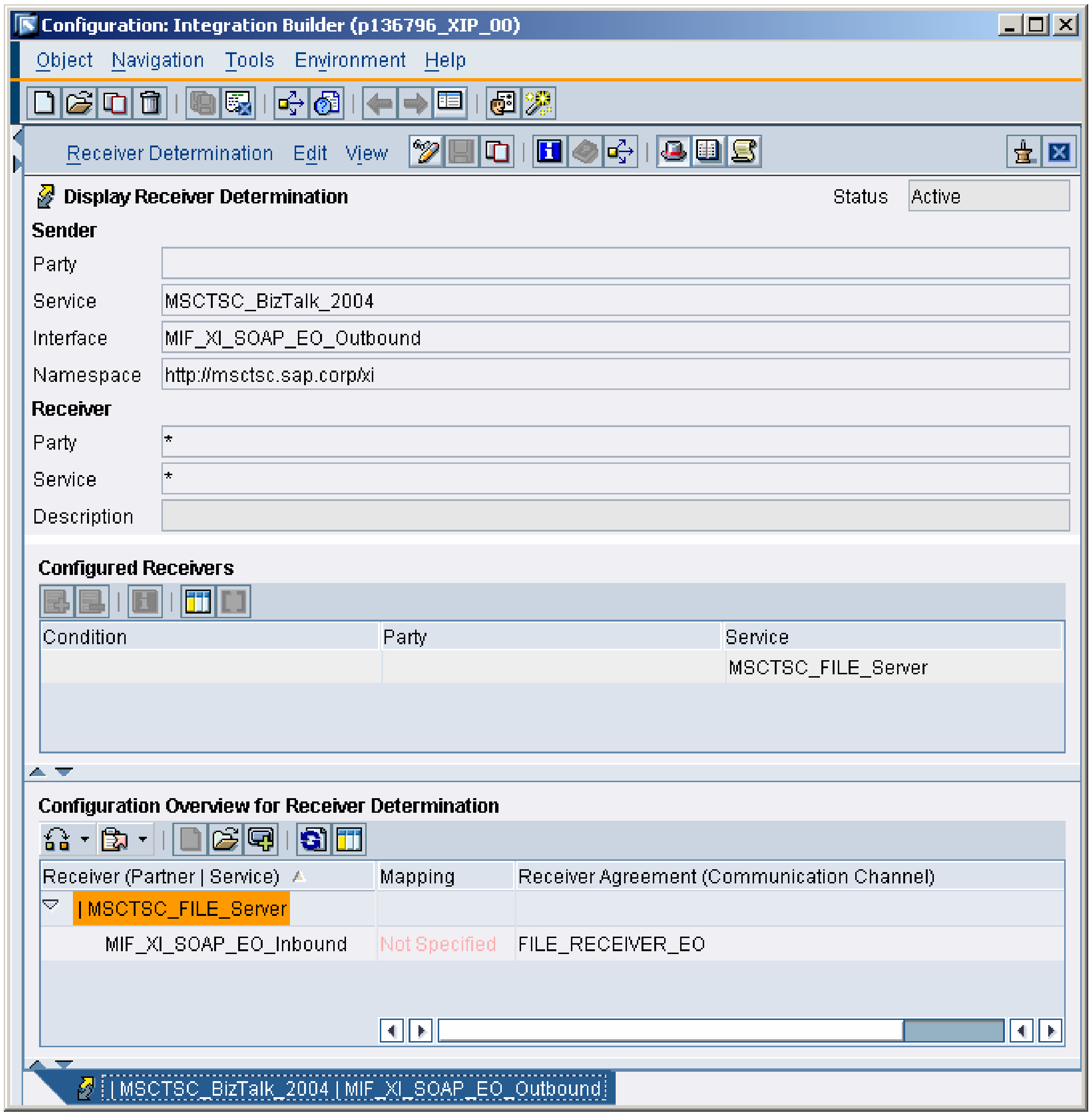
**Figure 40 SAP XI: SOAP Sender Adapter Configuration**

In order to eliminate duplicates for all cases, BizTalk sends the message with a unique message ID. This message ID will be used to create an XI message so that the identity of the created XI message and that of the original SOAP message in BizTalk are coupled. The MessageID will be passed as part of the HTTP query string:

```
http://host:port
/XISOAPAdapter/MessageServlet?channel=party:service:channel&
version=3.0&...&MessageId=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

where xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx is a GUID string

In our sample scenario the message coming from BizTalk is routed to a file server:


Configuration: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Receiver Determination Edit View

Display Receiver Determination — Status: Active

**Sender**

| | |
|---|---|
| Party | |
| Service | MSCTSC_BizTalk_2004 |
| Interface | MIF_XI_SOAP_EO_Outbound |
| Namespace | http://msctsc.sap.corp/xi |

**Receiver**

| | |
|---|---|
| Party | * |
| Service | * |
| Description | |

Configured Receivers

| Condition | Party | Service |
|---|---|---|
| | | MSCTSC_FILE_Server |

Configuration Overview for Receiver Determination

| Receiver (Partner \| Service) | Mapping | Receiver Agreement (Communication Channel) |
|---|---|---|
| \| MSCTSC_FILE_Server | | |
| MIF_XI_SOAP_EO_Inbound | Not Specified | FILE_RECEIVER_EO |

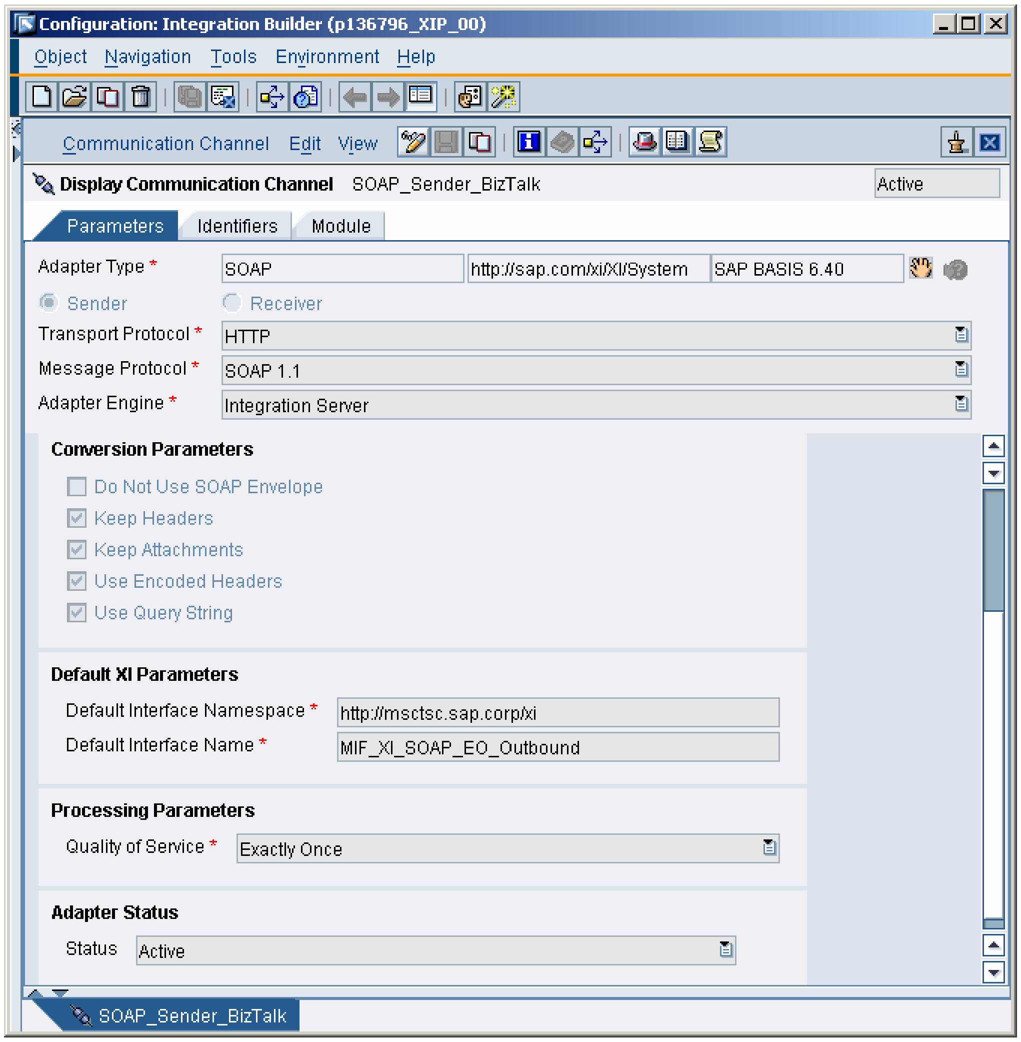
| MSCTSC_BizTalk_2004 | MIF_XI_SOAP_EO_Outbound

**Figure 41 SAP XI: Receiver Determination**

At the moment the BizTalk part is setup without any resend mechanism: BizTalk sends a SOAP message and ignores the response completely as in "fire-and-forget model". The Quality of Service with *AtMostOnce* is realized.

The next screenshot shows the configuration of the SOAP-Sender Adapter.

Note: Since the GUID is set by BizTalk, you must set the *Use Encoded Headers* and *Use Query String* indicators.


Configuration: Integration Builder (p136796_XIP_00)
Object Navigation Tools Environment Help
Communication Channel Edit View
Parameters Identifiers Module
SOAP_Sender_BizTalk

Display Communication Channel SOAP_Sender_BizTalk Active

Adapter Type * SOAP http://sap.com/xi/XI/System SAP BASIS 6.40

[✓ Sender] Receiver

Transport Protocol * HTTP

Message Protocol * SOAP 1.1

Adapter Engine * Integration Server

**Conversion Parameters**

- ☐ Do Not Use SOAP Envelope
- ☑ Keep Headers
- ☑ Keep Attachments
- ☑ Use Encoded Headers
- ☑ Use Query String

**Default XI Parameters**

Default Interface Namespace * http://msctsc.sap.corp/xi

Default Interface Name * MIF_XI_SOAP_EO_Outbound

**Processing Parameters**

Quality of Service * Exactly Once

**Adapter Status**

Status Active

**Figure 42 SAP XI: SOAP Sender Adapter Configuration**

Outlook

When BizTalk sends a SOAP message and checks if the response is an HTTP 200 response message, the Quality of Service with *AtLeastOnce* could be realized. In this case, the BizTalk should resend the message until such a successful response is returned. When the message is successfully accepted by the adapter, an HTTP 200 response with an empty SOAP envelope is returned.

BizTalk should send the message with the same message ID until an HTTP 200 response is returned or an HTTP 500 response with SOAP fault

DuplicateMessageException. In either case, BizTalk could assume that the message is delivered exactly once.

Note: This enhancement of the basic scenario has not been implemented and tested so far.

## Outlook

With next major releases of Microsoft and the next major SAP NetWeaver Release an adapter less communication based on enhanced web services will be possible between SAP XI and Microsoft BizTalk.

## Conclusion

It has been shown that reliable messaging between SAP XI 3.0 and Microsoft BizTalk Server 2004 can be obtained using SOAP compliant communication using the existing adapters. The implementation can be done using standard tools of both products.

## Limitations

Reliable messaging using the scenario described above requires the usage of SAP's Business Process Engine and Orchestration on the BizTalk side.

## References

- SAP Online Help: "Configuring the Sender SOAP Adapter" http://help.sap.com/saphelp_nw04/helpdata/en/fc/5ad93f130f9215e10000000a155106/frameset.htm

- BizTalk Framework Disassembler Pipeline Component http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdk/htm/ebiz_prog_pipe_chuq.asp

- Press release *Microsoft and SAP Raise the Stakes for Web Services for the Enterprise* http://www.sap.com/company/press/press.aspx?PressID=2799

- Microsoft BizTalk Server: BizTalk Server Framework 2.0: Document and Message Specification http://www.microsoft.com/biztalk/techinfo/BizTalkFramework20.doc

- Microsoft Online Help: "Dynamically Setting the URI of a Consumed Web Service" ms-help://MS.VSCC.2003/BTS_2004/SDK/htm/ebiz_prog_webservices_ycgl.htm

- Assigning to Dynamic Ports http://msdn.microsoft.com/library/default.asp?url=/library/en-us/sdk/htm/ebiz_prog_orch_wwet.asp