

IBM WebSphere Business Integration Connect Enterprise
and
Advanced Editions



Enterprise Integration Guide

Version 4.2.2

IBM WebSphere Business Integration Connect Enterprise
and
Advanced Editions



Enterprise Integration Guide

Version 4.2.2

Notices:

Before using this information and the product it supports, be sure to read the general information under “Notices and Trademarks” on page 181.

29June2004

This edition of this document applies to IBM WebSphere Business Integration Connect Enterprise Edition (5724-E87) and Advanced Edition (5724-E75), version 4.2.2.

To send us your comments about IBM WebSphere Business Integration documentation, email doc-comments@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Copyright International Business Machines Corporation 2003, 2004. All rights reserved. US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

© **Copyright International Business Machines Corporation 2003, 2004. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

New in this release	vii
--------------------------------------	------------

About this book	ix
Audience	ix
Typographic conventions	ix
Related documents	ix

Part 1. Introduction to back-end integration **1**

Chapter 1. Planning for back-end integration **3**

Overview of back-end integration	3
Planning the back-end integration	5
What business protocol are you using?	5
Which packaging will you use?	10
Which message transport will you use?	18
How do you access your back-end application?	22
Message handling	23
Queued delivery.	23
Communication error handling.	23
Duplicate messages.	24
Creating the transport-protocol mechanism.	24
HTTP/S transport-protocol mechanism	25
JMS transport-protocol mechanism	25
Configuring Business Integration Connect	25
Sending documents to the back-end system	26
Receiving documents from the back-end system	31

Part 2. Integrating with WebSphere InterChange Server **35**

Chapter 2. Introduction to InterChange Server Integration. **37**

Planning for integration with InterChange Server.	37
InterChange Server versions that Business Integration Connect supports	38
Message transports that InterChange Server supports	38
Support for InterChange Server integration.	40
Configuring Business Integration Connect for InterChange Server	41
Providing support for outgoing documents.	41
Providing support for incoming documents	42
Configuring InterChange Server	44
Creating business object definitions	44
Creating the connectors	47
Creating the collaborations	48
Deploy the project	49
Handling documents with attachments	49
Performing the conversion	50
Setting up the environment for the Attachment data handler	55
Configuring the Attachment data handler	56
Creating attachment-related business object definitions	60

Chapter 3. Integrating InterChange Server over HTTP **67**

Using HTTP transport protocol with pre-4.2.2 ICS	67
Sending documents to pre-4.2.2 ICS through HTTP	67
Receiving documents from pre-4.2.2 ICS through HTTP	81
Creating business object definitions for pre-4.2.2 ICS over HTTP.	87

Creating pre-4.2.2 ICS artifacts for HTTP	95
Using HTTP transport protocol with v4.2.2 ICS	95
Components required for documents to v4.2.2 ICS over HTTP transport	96
Setting up the environment for HTTP transport with v4.2.2 ICS	99
Creating business object definitions for v4.2.2 ICS over HTTP	102
Creating v4.2.2 ICS artifacts for HTTP	109
Sending SOAP documents over HTTP/S	110
Components required for sending and receiving	110
How a community participant invokes a Web Service	111
How the Community Manager invokes a Web Service	111
Chapter 4. Integrating with InterChange Server over JMS	113
Components required for documents over JMS transport	113
Sending documents over JMS transport	114
Receiving documents over JMS transport	116
Setting up the environment for JMS transport	119
Configuring the JMS queues	119
Configuring the Adapter for JMS	120
Creating business object definitions for JMS	121
Creating the payload business-object structure for JMS	122
Creating JMS header information	122
Creating ICS artifacts for JMS	125
Creating the JMS connector object	126
Binding collaborations to communicate with Adapter for JMS	126
<hr/>	
Part 3. Integrating with other back-end systems	127
Chapter 5. Integrating with WebSphere Business Integration Message Broker	129
Planning for integration with Message Broker	130
Message Broker versions that Business Integration Connect supports	130
Message transports that Message Broker supports	130
Support for Message Broker integration	131
Configuring Business Integration Connect for Message Broker	131
Providing support for outgoing documents	131
Providing support for incoming documents	133
Configuring Message Broker	134
Creating the message flow	134
Deploy the project	135
Using HTTP transport protocol with Message Broker	135
Components required for documents over HTTP transport	135
Creating the message flow for HTTP transport	136
Sending SOAP documents	138
Using JMS transport protocol with Message Broker	139
Components required for documents over JMS transport	139
Setting up the environment for JMS transport	143
Creating the message flow for JMS transport	144
Chapter 6. Integrating with WebSphere Data Interchange	147
Introduction	147
Sending documents to WebSphere Data Interchange	147
Receiving documents from WebSphere Data Interchange	148
Sample scenario used in this chapter	149
Configuring your environment for message exchange	150
Configure WebSphere MQ communication	150
Configuring WebSphere Data Interchange	152
Setting up the JMS environment	156
Configuring Business Integration Connect Enterprise Edition	157
Configuring Business Integration Connect - Express	164
Configuring My Profile	164
Creating a participant for Partner One	165

Configuring the Partner One participant	165
Summary	166
Chapter 7. Routing EDI documents	167
Overview of EDI routing	167
Special considerations for AS packaging	168
Routing the inbound document	168
Routing the outbound document	168
Setting both IDs in the participant profile	169
<hr/>	
Part 4. Appendixes	171
Appendix. Configuring a JMS protocol with WebSphere MQ	173
Configuring the JMS-configuration directory	173
Creating the JMS queues	174
Creating the MQ queue manager	174
Creating the MQ channels and transmission queue	175
Creating the MQ JMS local queues	176
Creating the JMS bindings file	177
Creating the JMS target	178
Creating the JMS gateway	179
Notices and Trademarks.	181
Notices	181
Programming interface information	183
Trademarks and service marks	183
Index	185

New in this release

This section describes the new features of IBM WebSphere Business Integration Connect that are covered in this version of the *Enterprise Integration Guide*. With this update to the IBM WebSphere Business Integration Connect 4.2.2 release, the following changes as been made to this document:

- The version 4.2.1 *Integration Guide* has been renamed as the *Enterprise Integration Guide*.
- The document has been extensively reorganized to provide better usability. In particular:
 - Information on how to configure WebSphere Business Integration Connect has been separated from information on how to configure a back-end system based on the assumption that different people or roles usually perform these two tasks.
 - Information on how to integrate with WebSphere InterChange Server has been expanded and broken into chapters, which are not contained in a Part 2 of the book. For an introduction to integration with InterChange Server, see Chapter 2, “Introduction to InterChange Server Integration,” on page 37.
- WebSphere Business Integration Connect can now use the WebSphere Business Integration Adapter for HTTP to provide support for integration with version 4.2.2 WebSphere InterChange Server over the HTTP transport protocol. For more information, see “Using HTTP transport protocol with v4.2.2 ICS” on page 95.
- A new chapter, Chapter 5, “Integrating with WebSphere Business Integration Message Broker,” on page 129, provides information on how to integrate WebSphere Business Integration Connect with WebSphere Business Integration Message Broker.

About this book

This document describes the Backend Integration interface, which is the mechanism that back-end systems and IBM^(R) WebSphere^(R) Business Integration Connect use to communicate. The document then describes how to integrate WebSphere InterChange Server and WebSphere Data Interchange with Business Integration Connect using the Backend Integration interface.

Audience

This book is intended for the person responsible for integrating Business Integration Connect Enterprise or Advanced edition with back-end systems.

Typographic conventions

This document uses the following conventions.

courier font	Indicates a literal value, such as a command name, filename, information that you type, or information that the system prints on the screen.
bold	Indicates a new term the first time that it appears.
<i>italic, italic</i>	Indicates a variable name or a cross-reference.
<i>blue outline</i>	A blue outline, which is visible only when you view the manual online, indicates a cross-reference hyperlink. Click inside the outline to jump to the object of the reference.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
[]	In a syntax line, square brackets surround an optional parameter.
...	In a syntax line, ellipses indicate a repetition of the previous parameter. For example, <code>option[...]</code> means that you can enter multiple, comma-separated options.
< >	In a naming convention, angle brackets surround individual elements of a name to distinguish them from each other, as in <code><server_name><connector_name>tmp.log</code> .
/, \	In this document, backslashes (\) are used as the convention for directory paths. For UNIX installations, substitute slashes (/) for backslashes. All IBM WebSphere InterChange Server product pathnames are relative to the directory where the IBM WebSphere InterChange Server product is installed on your system.
%text% and \$text	Text within percent (%) signs indicates the value of the Windows text system variable or user variable. The equivalent notation in a UNIX environment is <code>\$text</code> , indicating the value of the <code>text</code> UNIX environment variable.
<i>ProductDir</i>	Represents the directory where the product is installed.

Related documents

The complete set of documentation available with this product includes comprehensive information about installing, configuring, administering, and using WebSphere Business Integration Connect Enterprise and Advanced Editions.

You can download this documentation or read it directly online at the following site: <http://www.ibm.com/software/integration/wbiconnect/library/infocenter>

Important information about this product may be available in Technical Support Technotes and Flashes issued after this document was published. These can be found on the WebSphere Business Integration Support Web site at: <http://www.ibm.com/software/integration/websphere/support>. Select the component area of interest and browse the Technotes and Flashes section.

Part 1. Introduction to back-end integration

Chapter 1. Planning for back-end integration

This chapter describes how to plan for integration of IBM WebSphere Business Integration Connects with a back-end system. Within the hub community, you exchange business documents with your community participants. The purpose of exchanging these documents is to communicate information, which typically involves processing data and returning a result. When you receive data from a community participant, processing of that data generally occurs in the back-end system of your enterprise.

This chapter provides the following general information about back-end integration:

- “Overview of back-end integration”
- “Planning the back-end integration” on page 5
- “Message handling” on page 23

Overview of back-end integration

With Business Integration Connect, you exchange business documents with your community participants. The purpose of exchanging these documents is to communicate information, which typically involves processing data and returning a result. When you receive data from a community participant, processing of that data generally occurs in the back-end system of your enterprise. WebSphere Business Integration Connect is the point within the hub community through which messages to and from the enterprise are routed.

For communication with the enterprise, the following components of the trading community are involved:

- The **Community Manager** is the participant that handles the sending and receiving of messages from community participants to the back-end system. The Community Manager uses WebSphere Business Integration Connect Enterprise or Advanced Edition, as follows:
 - Business Integration Connect Enterprise edition acts as a Community Manager of the enterprise.
 - Business Integration Connect Advanced edition can act as a Community Manager of its own small hub community and a community participant of a larger hub community.

The Hub Administrator (Hub Admin) designates one participant in the hub community as the Community Manager.

- The **back-end system** manages messages between the Community Manager and your enterprise, as follows:
 - Messages can be sent *from* your enterprise to the Community Manager by going through the back-end system. In this case, the back-end system generates the document. Business Integration Connect detects this document and routes it to the appropriate community participant.
 - Messages can be routed *to* your enterprise by going through the Community Manager to the back-end system. In this case, a community participant generates the document. Business Integration Connect detects the document and routes it to the back-end system, which processes the document, perhaps routing it to other destinations within the enterprise.

The enterprise is accessed through a back-end system to which Business Integration Connect connects. All editions of Business Integration Connect provide the ability to connect to back-end systems. These editions differ in the transport protocols they can support, as follows:

- Business Integration Connect - Express provides file-based integration.
- Business Integration Connect Enterprise and Advanced editions provide file-based integration. In addition, they provide integration over the HTTP, HTTPS, and JMS protocols.

Documents exchanged between the community participant and Business Integration Connect can be in a variety of formats, in addition to RosettaNet. Documents can be in the SOAP, cXML, XML, EDI, or binary formats. The *Administrator Guide* has a complete list of the document types supported as well as the transport protocols (for example, HTTP) that can be used to send the documents.

Consider the following example--a community participant sends a RosettaNet-formatted purchase order, intended for the Community Manager, to the appropriate target on Business Integration Connect (Enterprise or Advanced edition). The Community Manager has a back-end system that processes purchase orders and expects to receive the orders in RosettaNet Service Content (RNSC) format. When the connection between the community participant and Community Manager is established, it is agreed that:

- The document will be translated from RosettaNet to RNSC format.
- The document routed to the back-end system will have Backend Integration packaging, meaning that transport-level headers will be added to the document to convey information needed for the exchange.

The back-end system can then process the document.

Documents that can be exchanged between Business Integration Connect and the back-end system of the Community Manager as well as the transport types associated with the documents are shown in Table 15 on page 27 and Table 20 on page 32..

Figure 1 illustrates how Business Integration Connect uses the back-end integration interface to communicate with the back-end system at the Community Manager. Note that the arrows go in both directions; that is, the document can originate from the back-end system of the Community Manager.

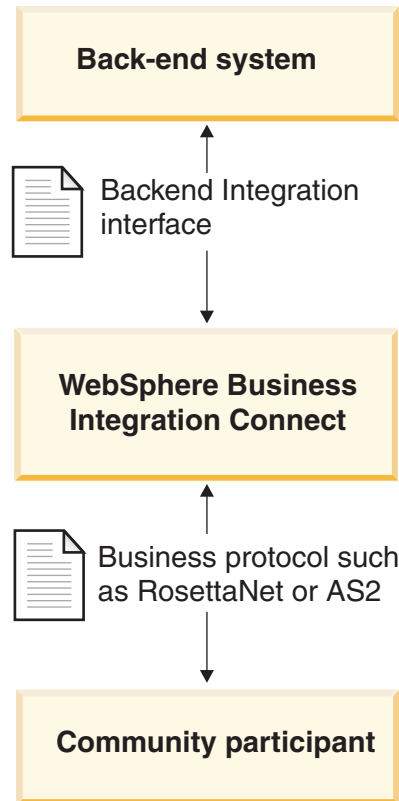


Figure 1. The role of the business protocol and packaging in the flow of documents

Planning the back-end integration

This section provides the following information about how to plan your back-end integration with WebSphere Business Integration Connect:

- “What business protocol are you using?”
- “Which packaging will you use?” on page 10
- “Which message transport will you use?” on page 18
- “How do you access your back-end application?” on page 22

What business protocol are you using?

The business protocol of your message determines the format of the document format. The business protocol affects many of the decisions you must make as you plan for integration to a back-end system. The choice of business protocol determines the packaging method you must use, which in turn, affect which message-transport protocols you can use.

For a complete description of business protocols, see the *Administrator Guide*. This section describes integration information that is specific to the following business protocols:

- “Web Services (SOAP)” on page 6
- “cXML” on page 6
- “RosettaNet” on page 7

Web Services (SOAP)

Business Integration Connect can make the following Web Services available to members of the hub:

- Web Services provided by the Community Manager can be available to community participants.

You will have to provide your community participant with the public WSDL that Business Integration Connect generates. It is important to note that the URL on which the community participant invokes the Web service is the Web Service Public URL specified while uploading the Web Service. Business Integration Connect acts as a proxy. It receives a SOAP message from the participant and figures out the corresponding private Web Service. It then invokes the private Web Service (provided by the Community Manager) using the same SOAP message. The response returned by the Community Manager is then returned to the participant.

- Web Services provided by community participants can be available to the Community Manager.

It is important to note that the same Web Service Interface can be provided by multiple partners. Business Integration Connect makes the Web Service available to the Community Manager at the Web Service URL specified in the console while uploading the Web Service. Additionally the Community Manager will have to provide the URL parameter to identify "To Partner". Refer to the *Hub Configuration Guide* for more details. Business Integration Connect acts as a proxy. It receives a SOAP message from the Community Manager and figures out the corresponding Web service and the "To Partner". It then invokes the Web Service provided by the partner using the same SOAP message. The response message returned by the partner is then returned to the Community Manager.

Refer to the *Hub Configuration Guide* for more information, including how to set up your document flow definitions for Web Services.

cXML

You can send or receive cXML documents to or from your community participants. When Business Integration Connect receives a cXML document from a community participant, it validates the document and translates it (if specified) before sending it to the back-end system at the Community Manager. Note that translation should not be used for synchronous cXML messages. In a synchronous exchange, the back-end system generates a response, which Business Integration Connect returns to the community participant (if appropriate for the message).

A back-end system at the Community Manager that needs to send a cXML document can do one of two things:

- Generate and send a cXML document, which Business Integration Connect passes through to the community participant
- Generate and send an XML document, which Business Integration Connect converts to cXML before sending to the community participant

Note: If XML document translation is used, for synchronous request/response transactions with the community participant, the response is returned asynchronously to the back-end system.

Refer to the *Administrator Guide* for more information, including how to set up your document flow definitions for cXML.

RosettaNet

Business Integration Connect provides support for RosettaNet 1.1 and 2.0, assuming the RosettaNet messages have Backend Integration packaging (that is, they must have transport-level headers.) These messages must use the HTTP or JMS transport protocol. The transport-level header retains meta-information that is not part of the PIP and enables Business Integration Connect to route the message appropriately.

For example, suppose an application wants to send a message to a community participant using RosettaNet sent on HTTP. The application provides the RosettaNet service content and adds the transport-level header. The header identifies which community participant will handle the request, what PIP will be sent, and the version of the PIP along with other information. This information enables Business Integration Connect to send the correct PIP to the community participant.

You can find information about setting up RosettaNet support and configuring PIPS in the *Hub Configuration Guide*.

Event notification: Because Business Integration Connect separates the application from the community participant that is the RosettaNet service provider, Business Integration Connect provides event notification. **Event notification** enables Business Integration Connect to, for example, notify the application if Business Integration Connect is unable to send a PIP to the participant. The application can then handle the failure.

An event notification message is an XML document that carries information about events that occurred within Business Integration Connect or an application. These messages have the same structure as any other message that enters or leaves Business Integration Connect; that is, they have transport-level header and payload. Business Integration Connect can be configured to send or not send event notification messages as they are optional.

Table 1 summarizes the event notification messages that Business Integration Connect can send to back-end systems.

Table 1. Event notification messages sent to back-end system

Event condition	Event notification message
Business Integration Connect delivers a RosettaNet document to a community participant and receives a Receipt Acknowledgment.	Event 100
Business Integration Connect cancels a PIP by generating an 0A1 message and delivering it to the community participant.	Event 800
Business Integration Connect receives a Receipt Acknowledgment exception or a general exception from a community participant.	Event 900

Business Integration Connect can send 0A1 message to the destination application as it would do with any other PIP, if it has been configured to send these messages using Exclusion List Management. See “Managing Exclusion Lists” in the *Administrator Guide*.

An application can send a event notification message to Business Integration Connect to cancel a RosettaNet PIP.

Event message structure: An event notification message has the standard transport-level header with the x-aux-process-type field set to XMLEvent. However, the payload of the message has a specific structure, as shown in the sample XML schema in Figure 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace=
  "http://www.ibm.com/websphere/bcg/2003/v1.0/xmleventnotification"
  xmlns:evntf=
  "http://www.ibm.com/websphere/bcg/2003/v1.0/xmleventnotification"
  elementFormDefault="qualified">
  <!-- EventNotification version 1.0 document element -->
  <xsd:element name="EventNotification">
    <xsd:complexType>
      <xsd:all>
        <xsd:element ref="evntf:StatusCode"/>
        <xsd:element ref="evntf:StatusMessage"/>
        <xsd:element ref="evntf:EventMessageID"/>
        <xsd:element ref="evntf:BusinessObjectID"/>
        <xsd:element ref="evntf:GlobalMessageID"/>
        <xsd:element ref="evntf:Timestamp"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
  <!-- StatusCode element -->
  <xsd:element name="StatusCode">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="100"/>
        <xsd:enumeration value="800"/>
        <xsd:enumeration value="900"/>
        <xsd:enumeration value="901"/>
        <xsd:enumeration value="902"/>
        <xsd:enumeration value="903"/>
        <xsd:enumeration value="904"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <!-- StatusMessage element -->
  <xsd:element name="StatusMessage">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
  </xsd:element>
```

Figure 2. Sample XML schema for an event notification message (Part 1 of 2)

```

<!-- EventMessageID element -->
  <xsd:element name="EventMessageID">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
  </xsd:element>
<!-- BusinessObjectID element -->
  <xsd:element name="BusinessObjectID">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
  </xsd:element>
<!-- GlobalMessageID element -->
  <xsd:element name="GlobalMessageID">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
  </xsd:element>
<!-- Timestamp element -->
  <xsd:element name="Timestamp">
    <xsd:simpleType>
      <xsd:restriction base="xsd:dateTime"/>
    </xsd:simpleType>
  </xsd:element>
</xsd:schema>

```

Figure 2. Sample XML schema for an event notification message (Part 2 of 2)

Table 2 describes each field within the event payload.

Table 2. Event notification XML fields

Field	Description
StatusCode	Type of message. The valid values are: <ul style="list-style-type: none"> • 100 - Business Integration Connect has delivered the document and received a receipt acknowledgment. • 800 - The application cancelled the PIP. • 900 - Business Integration Connect received a Receipt Acknowledgment exception, a general exception, or a 0A1Failure PIP from the community participant.
StatusMessage	Alpha-numeric description of this event notification message
EventMessageID	Alpha-numeric identifier of this particular event notification message
BusinessObjectID	The x-aux-msg-id in the transport-level header of the message affected by this message notification event. This links the payload of the original message to this event.
GlobalMessageID	The x-aux-system-msg-id in the transport-level header of the message that caused this message notification event
Timestamp	When the event occurred using the UTC time stamp format: CCYY-MM-DDThh:mm:ssZ including fractional precision of seconds (...ss.ssssZ). The date stamp must conform to the XML schema data type for dateTime (w3.org/TR/2001/REC-xmlschema-2-20010502#dateTime)

Event notification message example: Figure 3 shows an example of an event notification message sent using the HTTP protocol.

```

POST /builderURL HTTP/1.1
Content-Type: application/xml
Content-length: 250
x-aux-sender-id: 000000001
x-aux-receiver-id: 000000002
x-aux-third-party-bus-id: 000000003
x-aux-create-datetime: 2002-10-28T23:05:02Z
x-aux-protocol: XMLEvent
x-aux-protocol-version: 1.0
x-aux-process-type: XMLEvent
x-aux-process-version: 1.0
x-aux-payload-root-tag: evtmf:EventNotification
x-aux-msg-id: 98732
x-aux-system-msg-id: 12345
x-aux-production: Production
x-aux-process-instance-id: 3456
x-aux-event-status-code: 100
x-aux-transport-retry-count: 0

<?xml version="1.0" encoding="UTF-8"?>
<evntmf:EventNotification xmlns:evntmf=
"http://www.ibm.com/websphere/bcg/2003/v1.0/xmleventnotification">
  <evntmf:StatusCode>100</evntmf:StatusCode>
  <evntmf:StatusMessage>The message was delivered</evntmf:StatusMessage>
  <evntmf:EventMessageID>12345</evntmf:EventMessageID>
  <evntmf:BusinessObjectID>34234</evntmf:BusinessObjectID>
  <evntmf:GlobalMessageID>98732</evntmf:GlobalMessageID>
  <evntmf:Timestamp>2001-01-31T13:20:00Z</evntmf:Timestamp>
</evntmf:EventNotification>

```

Figure 3. Example of an event notification message using HTTP

Which packaging will you use?

The packaging type determines the format in which Business Integration Connect sends the message to the back-end system.

You use the Community Console to establish the connection with your community participants and to specify the packaging that is used between Business Integration Connect and the back-end system. To determine which packaging to use, you must understand the following issues:

- Which packaging types are valid for use with a back-end system?
- Which packaging types are valid with a message in a particular business protocol?

For more information on how to set up partner connections, see the *Hub Configuration Guide*.

Valid packaging types for integration

Not all packaging types are valid when you use Business Integration Connect for integration. Table 3 lists the packaging types that are relevant when Business Integration Connect is acting as a Community Manager.

Table 3. Packaging types relevant for back-end integration

Packaging type	Description
None packaging	Causes Business Integration Connect to send the message to the back-end system <i>without</i> any header data

Table 3. Packaging types relevant for back-end integration (continued)

Packaging type	Description
Backend Integration packaging	Adds additional attributes to the message header and, optionally, wraps the message contents in an XML transport envelope

Note: Other packaging types (such as AS) are available with Business Integration Connect. However, for integration with back-end systems, only the None and Backend Integration packaging types are recommended.

None packaging: When packaging is set to None, Business Integration Connect neither adds a transport-level header when it sends a message to a back-end system nor expects one when it receives a message from a back-end system. Instead, Business Integration Connect sends only the message to the back-end system. Information in the document controls routing.

Backend Integration packaging: When packaging is set to Backend Integration, messages sent to or received from a back-end system must have the following components:

- A transport-level header, which contains meta-information about the message
- A payload, which contains the content of the message
- An attachment (optional)

The header and payload are mandatory while attachments are optional. The following sections describe each of the components of a document that uses Backend Integration packaging.

Transport-level header content: The transport-level header contains information that Business Integration Connect uses to process and route the message to the correct destination. The transport-level header is bi-directional so that all messages entering and leaving Business Integration Connect have the mandatory fields and any of the optional fields that apply.

Table 4 lists the fields of the transport-level header.

Table 4. Transport-level header fields

Header field	Description	Required?
x-aux-sender-id	Identifier of the message sender, such as a DUNS number.	Yes
x-aux-receiver-id	Identifier of the message receiver, such as a DUNS number.	Yes
x-aux-protocol	Protocol of the message content. Valid values include RNSC for RosettaNet service content, XMLEvent, and Binary. For Business Integration Connect, the value in this field has priority over any protocol field in the payload.	Yes
x-aux-protocol-version	Version of the message content protocol.	Yes
x-aux-process-type	Process to be performed or what type of message is being sent. For RosettaNet messages, this is the PIP code such as 3A4. For event messages, it is XMLEvent and for Binary messages, it is Binary. For Business Integration Connect, the value in this field has priority over any process field in the payload.	Yes
x-aux-process-version	Version of the process. For RosettaNet messages, this is the version number of the PIP.	Yes
x-aux-create-datetime	When the message was successfully posted using the UTC time stamp format (CCYY-MM-DDThh:mm:ssZ)	

Table 4. Transport-level header fields (continued)

Header field	Description	Required?
x-aux-msg-id	Identifier of the payload content. For example, it could be the identifier of the RNPIPServiceContent instance for a RosettaNet message or it could be a proprietary document identifier. This links the message payload with something in the message sender's system for tracing purposes.	
x-aux-production	Routing of the message. Valid values are: Production Test This value is populated for requests in both directions. Note that when the message is a response to a two-way PIP initiated by a community participant, Business Integration Connect uses the GlobalUsageCode in the request and ignores the value in the transport level header.	
x-aux-system-msg-id	Global Unique Identifier (GUID) for the message, which is used for duplicate checking.	Yes
x-aux-payload-root-tag	Root tag element of the payload. For example, for 3A4 RosettaNet service content, the value of this field would be Pip3A4PurchaseOrderRequest. For event notification messages, the value for this field would be EventNotification.	
x-aux-process-instance-id	Identifier that links documents in a multiple message business process to a unique process instance. For RosettaNet, it must be unique for RosettaNet processes within the last 30 days. All messages exchanged as part of a RosettaNet process instance, including retries, use the same process instance ID.	
x-aux-event-status-code	Status code for the event notification. See the StatusCode field in "Event message structure" on page 8.	
x-aux-third-party-bus-id	Identifier such as a DUNS number of the party that delivered the message. This can be different from both the x-aux-sender-id and the x-aux-receiver-id if a third party is hosting Business Integration Connect on behalf of the community owner.	
x-aux-transport-retry-count	Number of unsuccessful attempts at posting this message prior to this attempt. If a message posts successfully on the first attempt, the value of this field will be 0.	
content-type	The content type of the message.	
content-length	The length of the message (in bytes).	

Note: For compatibility with IBM WebSphere MQ (A JMS provider), the fields of a JMS protocol message use underscores instead of hyphens. For example, in a JMS message, there is an `x_aux_sender_id` field instead of an `x-aux-sender-id` field.

Table 4 provides an overview of the transport-level header information. The following sections provide transport-level header information specific to certain business protocols:

- "Transport-level header and a RosettaNet message"
- "Transport-level header and an AS2 message" on page 13
- "Transport-level header and an AS1 message" on page 15

Transport-level header and a RosettaNet message: Table 5 describes where Business Integration Connect obtains values for the fields of the transport-level header from a RosettaNet message.

Table 5. Transport-level header fields and RosettaNet content

Header field	Source of value: RosettaNet 2.0	Source of value: RosettaNet 1.1
x-aux-sender-id	<(DeliveryHeader)> <messageSenderIdentification> <PartnerIdentification> <GlobalBusinessIdentifier>	<ServiceHeader> <ProcessControl> <TransactionControl> <ActionControl> or <SignalControl> <PartnerRouter> <fromPartner> <PartnerDescription> <BusinessDescription> <GlobalBusinessIdentifier>
x-aux-receiver-id	<(DeliveryHeader)> <messageReceiverIdentification> <PartnerIdentification> <GlobalBusinessIdentifier>	<ServiceHeader> <ProcessControl> <TransactionControl> <ActionControl> or <SignalControl> <PartnerRouter> <toPartner> <PartnerDescription> <BusinessDescription> <GlobalBusinessIdentifier>
x-aux-protocol	Set value for RosettaNet: RNSC	Same as for RosettaNet 2.0
x-aux-protocol-version	Set value: 1.0	Same as for RosettaNet 2.0
x-aux-process-type	The source XPath is: /ServiceHeader/ProcessControl/ pipCode/GlobalProcessIndicatorCode	The source XPath is: /ServiceHeader/ProcessControl/ ProcessIdentity/GlobalProcessIndicatorCode
x-aux-process-version	The source XPath is: /ServiceHeader/ProcessControl/ pipVersion/VersionIdentifier	The source XPath is: /ServiceHeader/ProcessControl/ ProcessIdentity/VersionIdentifier
x-aux-payload-root-tag	The value of the version identifier of each PIP is in its PIP specification. Name of the PIP such as Pip3A4PurchaseOrderRequest	The value of the version identifier of each PIP is in its PIP specification. Same as for RosettaNet 2.0
x-aux-process-instance-id	For processes initiated by the application, the value is the ID of the process instance. For processes initiated by a community participant that are not pass-through workflow, the value is the process ID in the initial RosettaNet request: <ServiceHeader> <ProcessControl> <pipInstanceId> <InstanceIdentifier>	<ServiceHeader> <ProcessControl> <ProcessIdentity> <InstanceIdentifier>
x-aux-msg-id	<(RNPipServiceContent)> <thisDocumentIdentifier> <ProprietaryDocumentIdentifier>	Same as for RosettaNet 2.0
x-aux-production	<ServiceHeader> <ProcessIndicator> <GlobalUsageCode>	<Preamble> <GlobalUsageCode>

Transport-level header and an AS2 message: Table 6 describes where Business Integration Connect obtains values for the fields of the transport-level header from an AS2 message.

Note: The values are case-sensitive.

Table 6. Transport-level header fields from AS2 content

Header field	Source of value
x-aux-sender-id	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the AS2-From header field of the AS2 message is set in the x-aux-sender-id field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-sender-id field of the incoming back-end integration message is used as the AS2-From header value of the AS2 message.
x-aux-receiver-id	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the AS2-To header field of the AS2 message is set in the x-aux-receiver-id field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-receiver-id field of the incoming back-end integration message is used as the AS2-To header value of the AS2 message.
x-aux-protocol	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the ToProtocol of the participant connection is set in the x-aux-protocol field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-protocol field of the incoming back-end integration message is used to determine the FromProtocol of the participant connection.
x-aux-protocol-version	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the ToProtocolVersion of the participant connection is set in the x-aux-protocol-version field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-protocol-version field of the incoming back-end integration message is used as the FromProtocolVersion of the participant connection.
x-aux-process-type	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the ToProcessCode of the participant connection is used to set in the x-aux-process-type field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-process-type field of the incoming back-end integration message is used as the FromProcessCode of the participant connection.
x-aux-process-version	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, the ToProcessVersion of the participant connection is set in the x-aux-process-version field of the back-end integration message that is sent to the Community Manager. When an AS2 message goes out to a community participant, the x-aux-process-version field of the incoming back-end integration message is used as the FromProcessVersion of the participant connection.
x-aux-payload-root-tag	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, for custom XML protocol only, the Root tag specified in the XPATH is parsed out of the message and used in the x-aux-payload-root-tag field. When an AS2 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.
x-aux-process-instance-id	This field is not used for AS2.
x-aux-msg-id	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, for custom XML protocol only, the Doc ID specified in the XPATH is parsed out of the message and used in the x-aux-payload-root-tag field. When an AS2 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.
x-aux-system-msg-id	When a community participant sends an AS2 message to Business Integration Connect Enterprise or Advanced edition, this field is set to the internally generated unique ID for this message. When an AS2 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.
x-aux-production	This field is not used for AS2.

Transport-level header and an AS1 message: Table 7 describes where Business Integration Connect obtains values for fields in the transport-level header from an AS1 message.

Note: The values are case-sensitive.

Table 7. Transport-level header fields from AS1 content

Header field	Source of the value
x-aux-sender-id	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, the <i>FromID</i> in the "Subject: <i>ToID;FromID</i> " header field of the AS1 message is set in the x-aux-sender-id field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-sender-id field of the incoming back-end integration message is used as <i>FromID</i> in the "Subject: <i>ToID;FromID</i> " header value of the AS1 message.
x-aux-receiver-id	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, <i>ToID</i> in the "Subject: <i>ToID;FromID</i> " header field of the AS1 message is set in the x-aux-receiver-id field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-receiver-id field of the incoming back-end integration message is used as <i>ToID</i> in the "Subject: <i>ToID;FromID</i> " header value of the AS1 message.
x-aux-protocol	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, the <i>ToProtocol</i> of the participant connection is set in the x-aux-protocol field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-protocol field of the incoming back-end integration message is used as the <i>FromProtocol</i> of the participant connection.
x-aux-protocol-version	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, the <i>ToProtocolVersion</i> of the participant connection is set in the x-aux-protocol-version field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-protocol-version field of the incoming back-end integration message is used as the <i>FromProtocolVersion</i> of the participant connection.
x-aux-process-type	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, the <i>ToProcessCode</i> of the participant connection is set in the x-aux-process-type field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-process-type field of the incoming back-end integration message is used as the <i>FromProcessCode</i> of the participant connection.
x-aux-process-version	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, the <i>ToProcessVersion</i> of the participant connection is set in the x-aux-process-version field of the back-end integration message that is sent to the Community Manager. When an AS1 message goes out to a community participant, the x-aux-process-version field of the incoming back-end integration message is used as the <i>FromProcessVersion</i> of the participant connection.
x-aux-payload- root-tag	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, for custom XML protocol only, the Root tag specified in the XPATH is parsed out of the message and set in the x-aux-payload-root-tag field. When an AS1 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.
x-aux-process-instance-id	This field is not used for AS1.
x-aux-msg-id	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, for custom XML protocol only, the Doc ID specified in the XPATH is parsed out of the message and used in the x-aux-payload-root-tag field. When an AS1 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.

Table 7. Transport-level header fields from AS1 content (continued)

Header field	Source of the value
x-aux-system-msg-id	When a community participant sends an AS1 message to Business Integration Connect Enterprise or Advanced edition, this field is set to the internally generated unique ID for this message. When an AS1 message goes out to a community participant, this field doesn't need to be set in the incoming back-end integration message.
x-aux-production	This field is not used for AS1.

Payload: The payload of the message contains the actual content of the message. The location of the payload depends on the transport protocol that is sending the message, as Table 8 shows.

Table 8. Location of payload

Transport protocol	Location of payload
HTTP protocol messages	In the body of the HTTP post
JMS protocol messages	In the body of the JMS message
RosettaNet messages	The service content from the PIP
Sending EDI over AS2	The EDI message
	The payload is <i>not</i> wrapped with an XML envelope unless the message also carries one or more attachments. For information on the XML envelope and tags used to wrap the attachments, see "Attachments" on page 17.

The payload can be Base64-encoded and in an XML **transport envelope** in either of the following cases:

- If the document contains an attachments
 - A document with attachments *must* be wrapped in an XML transport envelope. For more information on how attachments are formatted, see "Attachments" on page 17.
- If you set the envelope flag for Backend Integration packaging to Yes
 - To wrap a document in the XML transport envelope *regardless* of whether it contains attachments, set the Backend Integration envelope flag to Yes from the profile's B2B Capabilities screen. For example, to set this flag in the Hub Operator's profile, choose:
 - Profile > Hub Operator > B2B Capabilities
 - Click on Edit for Backend Integration to see the Envelope Flag.

This XML transport envelope wraps the document in the <transport-envelope> root tag. Inside this root tag there is a <payload> tag that contains the document payload. If any attachments exist, each is contained in an <attachment> tag. For more information on the structure of these tags, see "Attachments" on page 17.

Business Integration Connect includes the following W3C XML schema file that describes the Backend Integration XML transport envelope structure:

wbipackaging_v1.0_ns.xsd

This schema file is located in the following directory on the installation medium:
B2BIntegrate\packagingSchemas

You can use any XML editing tool to validate your Backend Integration XML against this schema file to ensure the document is valid before it is sent to the Document Manager.

Attachments: If the business message protocol permits them, each document can have one or more attachments. If the document has attachments, it *must* be wrapped in an XML transport envelope, as described in “Payload” on page 16. Table 9 describes the XML attributes in the payload and attachment tags.

Table 9. XML attributes of the payload and attachment tags

XML attribute	Description	Required?
Content-Type	Identifies the MIME type/subtype, such as text/xml or image/gif.	Yes
Encoding	Identifies the encoding. Because the attachment and payload must be Base64-encoded, the only valid value for this attribute is "Base64".	No

Figure 4 shows an example of a document in an XML transport envelope that contains the payload and one attachment.

Note: The namespace in this example is required:

```
xmlns="http://www.ibm.com/websphere/bcg/2003/v1.0/wbipackaging"
```

```
<?xml version="1.0" encoding="utf-8"?>
<transport-envelope
  xmlns="http://www.ibm.com/websphere/bcg/2003/v1.0/wbipackaging">
  <payload encoding="base64" contentType="application/xml">
    ...base64 encoded XML message...
  </payload>
  <attachment encoding="base64" Content-Type="text/xml">
    ...base64 encoded XML attachment...
  </attachment>
</transport-envelope>
```

Figure 4. Sample XML transport envelope for payload and one attachment

Note: To process documents wrapped in the XML transport envelope with the WebSphere InterChange Server, Business Integration Connect provides the Attachment data handler. For more information, see “Handling documents with attachments” on page 49.

Which packaging type works with your documents?

Documents in certain business protocols can use only certain types of packaging. For example, a RosettaNet document can be processed *only* when a packaging of Backend Integration has been specified. See Table 15 on page 27 and Table 20 on page 32 for a complete list of which document types can be associated with which types of packaging.

Example of Backend Integration packaging over HTTP

Figure 5 shows an example of a message from Business Integration Connect to an application using the HTTP transport protocol. Note that the message does not contain an attachment.

```
POST /sample/receive HTTP/1.1
Host: sample.COM
Content-Type: application/xml
Content-Length: nnn
x-aux-sender-id: 000000001
x-aux-receiver-id: 000000002
x-aux-third-party-bus-id: 000000003
x-aux-create-datetime: 2002-10-28T23:05:02Z
x-aux-protocol: RNSC
x-aux-protocol-version: 1.0
x-aux-process-type: 3A4
x-aux-process-version: V02.00
x-aux-payload-root-tag: Pip3A4PurchaseOrderRequest
x-aux-msg-id: 1021358129419
x-aux-system-msg-id: 2
x-aux-production: Production
x-aux-process-instance-id: 123456
x-aux-transport-retry-count: 0
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Pip3A4PurchaseOrderRequest SYSTEM
  "3A4PurchaseOrderRequestMessageGuideline_v1_2.dtd">
<Pip3A4PurchaseOrderRequest>
  <PurchaseOrder>
    ...
  </PurchaseOrder>
  ...
  <thisDocumentIdentifier>
    <ProprietaryDocumentIdentifier>1021358129419
    </ProprietaryDocumentIdentifier>
  </thisDocumentIdentifier>
  <GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>
</Pip3A4PurchaseOrderRequest>
```

Figure 5. Sample message using HTTP transport protocol

Which message transport will you use?

When the back-end system and WebSphere Business Integration Connect send messages to one another, each must use the same message-transport protocol. The **message-transport protocol** defines the communication protocol in which the messages are sent.

Business Integration Connect communicates with a back-end system through its Backend Integration interface. Table 10. lists the transport protocols that this Backend Integration interface supports.

Table 10. Transport protocols supported by Business Integration Connect

Transport protocol	For more information
HTTP or HTTPS	"HTTP transport protocol" on page 19
File-system files	"File-system protocol for Enterprise and Advanced editions" on page 21
JMS	"JMS protocol" on page 20

See Table 15 on page 27 and Table 20 on page 32 for information on which transport protocols are valid for a particular combination of message content and Backend Integration packaging.

HTTP transport protocol

To send messages using an HTTP protocol, Business Integration Connect uses HTTP/S 1.1. To receive messages from back-end systems, Business Integration Connect supports both HTTP/S version 1.0 and 1.1.

The HTTP message can include the integration packaging attributes. Whether these attributes are included depends on the packaging type associated with the participant connection, as follows:

- If the participant connection specifies that the HTTP message includes Backend Integration packaging, the transport-level header of the HTTP message includes additional attributes containing information about the message, such as the protocol of the content, the ID of the message, and the sender of the message. For a complete list of the fields in the header, see “Transport-level header content” on page 11.
EDI, SOAP, and cXML messages must use None packaging.
- If the participant connection specifies None packaging, the HTTP message does *not* have these additional attributes, and Business Integration Connect parses the message to obtain this information.
RosettaNet messages must use Backend Integration packaging.

Note: XML messages can use either None or Backend Integration packaging. Binary messages received from the back-end system must have the Backend Integration packaging; however, the reverse is not true because Business Integration Connect supports sending binary messages to the application using either type of packaging.

Process: When HTTP or HTTPS messages are sent between Business Integration Connect and an application for asynchronous exchanges, the following steps occur:

1. The source system (Business Integration Connect or the back-end system) posts an HTTP message to the target system using a specific URL.
2. The target system receives the message and sends the protocol-level acknowledgment, HTTP 200 or 202, to signify the change in ownership. The source system ignores the body of this acknowledgment message. If an error occurs during this processing, the target system sends an HTTP 500 message back to the source system.
3. If Business Integration Connect is the target system (that is, when Business Integration Connect receives a message), it then persists the message and releases the connection to the source system.
4. The target system can then process the message asynchronously.

When the exchange is synchronous (for example, for a SOAP or cXML document), a response is returned along with the HTTP 200 message in the same HTTP connection.

Sending and receiving messages using the HTTP protocol: To send a message to Business Integration Connect using the HTTP protocol, a back-end system takes the following steps:

1. Create the message.

The Content-Type attribute in the transport-level header gives the encoding used for the message.

2. Package the message according to the packaging type set for the connection. For Backend Integration packaging, the back-end system adds the protocol header attributes that Business Integration Connect requires.
3. Post the message to the URL that Business Integration Connect uses to receive these messages.
4. If the exchange is synchronous, the back-end system waits to receive a response in the same connection that was used for the request.

To enable HTTP message exchange in this direction, use the Target Details screen of the Community Console to set up a target for inbound documents. For more information, see “Receiving documents from the back-end system” on page 31.

To receive a message from Business Integration Connect using the HTTP protocol, a back-end system takes the following steps:

1. Listen for a message at a particular URL.
2. When a message is received, process the message:
 - If the connection has None packaging, the back-end system must parse the message to determine how to handle it.
 - If the connection has Backend Integration packaging, the application can use the Backend Integration attributes to determine how to handle the message.
3. If the exchange is synchronous, the back-end system returns a response in the same connection used for the request.

To enable HTTP message exchange in this direction, use the Gateway screen of the Community Console to set up a gateway for outbound documents. For more information, see “Sending documents to the back-end system” on page 26.

JMS protocol

The JMS protocol is based on the Java Message Service (JMS) and transfers messages through transactional, persistent JMS queues provided by, for example, IBM WebSphere MQ. The JMS Protocol supports the following JMS message types:

- StreamMessage (as a byte array)
- BytesMessage (as a byte array)
- TextMessage

In JMS protocol, the sending system sends a JMS message to the receiving system using the Enqueue operation. The receiving system gets the message from the queue, persists the message, and then performs the Dequeue operation to remove the message from the queue. From this point forward, the receiving system can process the message asynchronously.

The JMS message can include integration packaging attributes. Whether these attributes are included depends on the packaging type associated with the participant connection, as follows:

- If the participant connection specifies that the JMS message includes Backend Integration packaging, the JMS message contains transport-level information (such as the protocol of the content, the ID of the message, and the sender of the message) as JMS properties within the message. For a complete list of the properties, see “Transport-level header content” on page 11.

Note: For compatibility with WebSphere MQ JMS, the properties in the JMS messages use underscores in the property names instead of hyphens. For example, in a JMS message, the property is `x_aux_system_msg_id` while

the equivalent HTTP header field will be `x-aux-system-msg-id`. When Business Integration Connect processes a JMS message, it converts the underscores to hyphens in these properties.

- If the participant connection specifies None packaging, the JMS message does *not* have these additional attributes.

With the exception of binary messages, Business Integration Connect supports sending and receiving JMS messages with either type of packaging. Binary messages received from an application must have the Backend Integration packaging. The reverse is not true because Business Integration Connect supports sending binary messages to the application using either type of packaging.

Sending messages using the JMS protocol: To send a message to Business Integration Connect using the JMS protocol, a back-end system takes the following steps:

1. Create the message.
The `content_type` header attribute sets the content type for the message and the `content_length` header attribute specifies the length of the message (in bytes).
2. Package the message according to the packaging type set for the connection.
For Backend Integration packaging, the application adds the required JMS header attributes.
3. Send the message to the JMS queue that the back-end system uses to send messages to Business Integration Connect.

To enable JMS message exchange in this direction, use the Target Details screen of the Community Console to set up a target for inbound documents. For more information, see “Receiving documents from the back-end system” on page 31.

Receiving messages using the JMS protocol: To receive a message from Business Integration Connect using the JMS protocol, a back-end system takes the following steps:

1. Listen for a message on the JMS queue.
2. When a message is received, process the message:
 - If the connection has None packaging, the back-end system must parse the message to determine how to handle it.
 - If the connection has Backend Integration packaging, the application can use the Backend Integration attributes to determine how to handle the message.

To enable JMS message exchange in this direction, use the Gateway screen of the Community Console to set up a gateway for outbound documents. For more information, see “Sending documents to the back-end system” on page 26.

File-system protocol for Enterprise and Advanced editions

The File System protocol enables Business Integration Connect to send messages by placing them in a defined directory structure. Business Integration Connect receives messages by reading them from the directory structure. The file-system protocol supports the following:

- Document types: EDI and XML documents
- Integration packaging: only the None packaging type; that is, the files cannot contain additional attributes.

Sending messages using the file-system protocol: To send a message to Business Integration Connect using the file-system protocol, an application should take the following steps:

1. Create the message file in a temporary directory.
2. Once the file is ready, move the file to the directory that Business Integration Connect polls.

To enable file-system message exchange in this direction, use the Target Details screen of the Community Console to set up a target for inbound documents. The target of the message determines the directory that Business Integration Connect polls. When you create a target, Business Integration Connect creates a Documents directory and its subdirectories for the target, as follows:

```
<doc_root>
  Documents
    Production
    Test
  <other destination types>
```

Business Integration Connect polls the Documents directories and their subdirectories regularly to detect message files. If it finds a message, Business Integration Connect persists the message and then deletes the message from the directory. Business Integration Connect then processes the message normally. See the *Hub Configuration Guide* for information on how to create a target.

Receiving messages using the file-system protocol: To receive messages using the file system protocol, an application should do the following:

1. Poll the appropriate directory for message files.

Note: Temporary files (those with extensions `.tmp` or `.tmp1`) should be ignored. The application must *not* pick up or delete these temporary files.

2. When there is a message, persist it.
3. Delete the message from the directory.
4. Process the message.

To enable file-system message exchange in this direction, use the Gateway screen of the Community Console to set up a gateway for outbound documents. Business Integration Connect places the message file in the Documents directory, which the gateway defines. By defining the destination directory according to the gateway, each participant connection can have a different directory. For information on gateways, see the *Hub Configuration Guide*.

How do you access your back-end application?

Business Integration Connect provides the ability to integrate with many different back-end applications. Usually, a back-end application is accessed through a back-end system, such as an integration broker. Integration to the back-end systems listed in Table 11 are covered in this guide.

Table 11. Supported back-end systems for Business Integration Connect

Back-end system	For more information
WebSphere InterChange Server	Chapter 2, "Introduction to InterChange Server Integration," on page 37
WebSphere Business Integration Message Broker	Chapter 5, "Integrating with WebSphere Business Integration Message Broker," on page 129

Table 11. Supported back-end systems for Business Integration Connect (continued)

Back-end system	For more information
WebSphere Data Interchange	Chapter 6, "Integrating with WebSphere Data Interchange," on page 147

Message handling

This section describes how Business Integration Connect handles the following situations that impact the delivery of messages:

- "Queued delivery"
- "Communication error handling"
- "Duplicate messages" on page 24

Queued delivery

Business Integration Connect posts information on all documents that it wants to send to a particular gateway into a queue. The Delivery Manager system processes these messages in the order the queue receives them (FIFO) and uses a thread for each message to send them. Note that if the gateway (for example, URL if the transport protocol is HTTP or JMS destination if the transport protocol is JMS) has been configured to be offline (see Communication error handling), the messages remain in the queue until the gateway is enabled (online). If the Delivery Manager receives an error in a thread, it stops other threads from attempting to deliver their messages. The Delivery Manager places these messages back into the queue until it is able to deliver the message that caused the error.

If the number of failed attempts exceeds the maximum number of attempts, the Document Manager places the message in a failed directory and then attempts to deliver the next message in the queue unless the gateway is offline.

Communication error handling

When Business Integration Connect is the sender and the application returns an error (for example, an HTTP Response message that is not a 200 or 202 message when using the HTTP protocol), Business Integration Connect may then retry to send the message again depending on how it has been configured for this particular gateway. Each gateway (URL in the case of HTTP) has the following options that affect the number of retries and how the messages are sent:

Table 12. Gateway configuration options

Configuration options	Description
Retry Count	How many document retries to attempt if an error is received
Retry Interval	Time interval between retry attempts
Online/Offline	Starts and stops delivery attempts
Number of Threads	Number of posting threads that will process messages per gateway

If Business Integration Connect is not configured to retry sending the message or if all delivery attempts fail, Business Integration Connect signals the problem by doing any or all of the following actions:

- Presenting the errors in various screens of the Community Console such as the Document Viewer and RosettaNet Process Viewer

- Sending an e-mail to appropriate people to notify them of the problem so that they can take appropriate actions, if an e-mail alert for the delivery failed event has been set up
- Creating an event document and then sending that document to receiver.

See “Managing gateway configurations” in the *Administrator Guide* for more information.

Duplicate messages

All messages sent to or received from Business Integration Connect must have a Global Unique Identifier (GUID). Business Integration Connect uses the GUID to detect duplicate messages. When Backend Integration packaging is used, each message carries its GUID in the transport-level header. For the HTTP protocol, for example, the GUID is carried in the `x-aux-system-msg-id` field (see “Transport-level header content” on page 11). The sender of the message generates the GUID. The file system protocol does not support checking for duplicate messages.

If the attempt to send a message results in an error, Business Integration Connect reuses the message’s GUID in each retry. If Business Integration Connect receives a message that contains a duplicate GUID, it returns a positive acknowledgment (for example, HTTP 200) but does not process the duplicate message.

Note: Business Integration Connect checks for duplicate messages at the RosettaNet process level if RosettaNet is being used. It also checks for duplicate messages if XML is being used.

Creating the transport-protocol mechanism

For Business Integration Connect and the back-end system to communicate, you must choose a transport protocol that both these back-end-integration components can support. Table 13 summarizes the information that this guide provides on supported transport protocols.

Table 13. How to get information on supported transport protocols

Back-end-integration component	For more information
Business Integration Connect	“Configuring Business Integration Connect” on page 25
Back-end system	Refer to the appropriate chapter of this guide, as listed in “How do you access your back-end application?” on page 22.

For these two back-end-integration components to communicate, you must ensure that a **transport-protocol mechanism** exists; that is, the appropriate software and hardware entities must exist for these two components to communicate over the desired transport protocol. This section summarizes the steps in the creation of a transport-protocol mechanism for the following transport protocols:

- HTTP/S
- JMS
- File-system

HTTP/S transport-protocol mechanism

To communicate over the HTTP or HTTPS transport protocol, Business Integration Connect and the back-end system require a URL address, one that each can access. Therefore, you must supply a URL address and provide it to each of these components. This address must have the following format:

```
bcgreceiver
```

JMS transport-protocol mechanism

To communicate over the JMS transport protocol, Business Integration Connect and the back-end system require a JMS queue for *each* direction of the communication. Therefore, you must take the following steps to supply the appropriate JMS queues:

- Configure your JMS environment.
- Create a queue manager and the required queues including the transmission queue, remote queue, receiver queue.
- Create a sender and receiver channel.

The JMS queue manager can exist on any machine, including the following:

- The machine where the back-end system resides
- The machine where the WebSphere Business Integration Connect resides

In addition, you can have a queue manager on *both* the machine where the back-end system resides and the machine where Business Integration Connect resides. In this case, use setup channels to tie the two queue managers together. Using this method, neither side needs to make client connections over the network.

Instructions for configuring a JMS transport-protocol mechanism using WebSphere MQ version 5.3 are provided in “Configuring a JMS protocol with WebSphere MQ,” on page 173.

Note: It is also possible to use LDAP or WebSphere Application Server as a JNDI provider.

Configuring Business Integration Connect

This section summarizes steps for configuring Business Integration Connect for use with a back-end system. These configuration steps assume that you have already configured the community participants in your hub community. In particular, this section assumes that the following configuration has already been performed:

- A **participant** for Community Manager has already been created; its participant **profile** has also been created.
- Community **participants** for the origin (or destination) of the documents have already been created.
- In the community participants, participant **profiles** for the Community Manager participant have been created.
- A **target** has been defined so that the Business Integration Connect’s Receiver of Community Manager can listen for incoming documents from the community participant over the appropriate transport protocol.
- **B2B capabilities** have been defined and enabled in the community participant (from which the document is received) so that the Community Manager expects documents from that source.

- **Participant connections** exist between the Community Manager and the community participants so that a participant and the Community Manager can receive (or send) a document.

Note: You should login as the Community Administrator.

For a complete description of how to configure Business Integration Connect to support a hub community, see the *Hub Configuration Guide*.

Once the community participants are configured, you must configure Business Integration Connect so that it can communicate with a back-end system. This section provides the following information to describe how to incorporate a back-end system into your hub community:

- “Sending documents to the back-end system”
- “Receiving documents from the back-end system” on page 31

Sending documents to the back-end system

To send a document to the back-end system, the Community Manager takes the following steps:

1. Receive a document from some community participant.
The Receiver retrieves this source document from a target that has been defined in Community Manager for incoming messages from the community participant and its associated transfer protocol. When sending a document to the back-end system, the source document is the document that is received from some *community participant*; it is therefore referred to as the **participant document**.
2. Convert the participant document to the destination document, which is in format that the back-end system requires.
The Business Integration Connect Document Manager performs this conversion to the destination document. When sending a document to the back-end system, the destination document is the document that is sent to the *back-end system*; it is therefore referred to as the **back-end document**.
3. Send the back-end document to the back-end system.
The Document Manager sends the back-end document through a gateway that has been defined in Community Manager for outgoing messages to the back-end system.

Therefore, for the Community Manager to be able to send a document to the back-end system, you must ensure that the configuration summarized in Table 14 has been performed within Business Integration Connect Enterprise or Advanced edition.

Table 14. Configuration steps to send documents to the back-end system

Configuration step	Business Integration Connect steps	For more information
1. Define where to send the document.	1. Create a gateway to the back-end system.	“Defining where to send the participant document” on page 27

Table 14. Configuration steps to send documents to the back-end system (continued)

Configuration step	Business Integration Connect steps	For more information
2. Define how to process the document.	2. Create document flow definitions for the source and destination formats.	“Defining how to process the participant document” on page 28
	3. Enable B2B capabilities for the document flow definition of the document sent to the back-end system.	
	4. Create a document flow definition interaction between the source and destination document flow definitions.	
3. Define how to connect to the back-end system.	5. Create a participant connection that sends documents to the back-end system.	“Defining how to connect to the back-end system” on page 30

Defining where to send the participant document

To send documents to the back-end system, the Community Manager must have a gateway defined. This **gateway** specifies the destination for the converted documents; that is, it specifies the location (as a URI) to which the Community Manager sends the back-end document. This location is the same one at which the back-end system listens for incoming messages. The gateway identifies the entrance point into the enterprise application layer (within the back-end system). Within Business Integration Connect, it is Document Manager that checks for a gateway. Once the Document Manager has processed the document, it sends the converted document to the back-end system at the location specified in the gateway.

To define a gateway within Business Integration Connect Enterprise or Advanced edition, click:

Account Admin > Profiles > Gateways

When you define the gateway, you specify the transport protocol that the Community Manager and back-end system both use to transfer the back-end document. As Table 15 shows, the choice of transport protocol depends on the format of the document. Its format includes its packaging type and business protocol, which are defined in its document flow definition.

Note: For more information on how to create a gateway in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

Table 15. Supported transport protocols from Business Integration Connect to back-end system

Packaging type	Business protocol	HTTP or HTTPS?	JMS?	File system?
Backend Integration	RosettaNet (RNSC)	Yes	Yes	No
	XML	Yes	Yes	No
	Binary	Yes	Yes	No
None		Yes	Yes	Yes
	EDI only	Yes	Yes	Yes
	cXML only	Yes	No	No
	SOAP only	Yes	No	No
	Binary	Yes	Yes	No

Note: The choice of transport protocol also depends on the transport protocols your particular back-end system supports. For more information, refer to the chapter in this guide for integrating your particular back-end system.

Once you have selected a valid transport protocol for your document, you can provide the other information you need to define for the gateway in the Gateways screen.

Defining how to process the participant document

For the Document Manager to be able to process the participant document, it must know the format to which it needs to convert this document; that is, it needs to know the format of the back-end document. As part of the back-end integration, you must ensure that the following entities are defined within your Business Integration Connect:

- Document flow definitions must exist to define the format of both the participant document and the back-end document.
- The Community Manager's B2B capabilities must include enablement of the back-end document's document flow definition as a destination (target).
- A document flow definition interaction must exist that brings together the participant document as the source and the back-end document as the destination.

Defining the document flow definitions: Each **document flow definition** defines how Business Integration Connect processes a particular document. It includes the packaging type and business protocol of the document. Business Integration Connect provides some predefined packaging types and protocol definitions. If these predefined formats correctly define your participant and back-end documents, you do not need to define any document flow definition. However, if the predefined formats do *not* adequately define your participant or back-end document, you must create a valid document flow definition for that document. To define a document flow definition within Business Integration Connect Enterprise or Advanced edition, you use the Manage Document Flow Definitions screen of Business Integration Connect. To access this screen, click:

```
Hub Admin > Hub Configuration > Document Flow Definition >  
Create Document Flow Definition
```

Note: For more information on predefined document flow definitions as well as how to create document flow definitions in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

For back-end integration, the packaging type of the back-end document *must* be one of the following:

- None packaging
- Backend Integration packaging

You must determine which of these packaging types applies, based on the business protocol of your document and the particular back-end system you are using. For information on packaging types with back-end systems, see "Which packaging will you use?" on page 10. For information on supported back-end systems, see "How do you access your back-end application?" on page 22.

Setting the B2B capabilities for sending: Before the Document Manager can convert the source document, it must determine whether it can handle the format of the desired destination document. To make this determination, the Document Manager checks its **B2B capabilities**, which define which document flow

definitions have been enabled. Supported document flow definitions have each of their component document types (such as packaging type, business protocol, and document) enabled. To enable a particular document flow definition, you use the B2B Capabilities screen of Business Integration Connect. To access this screen, click: Account Admin > Profiles > B2B Capabilities

Note: For more information how to set B2B capabilities in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

For back-end integration, make sure each of the component document types for the back-end document's document flow definition has been enabled to serve as a destination (target). Table 16 summarizes the action to take on the B2B Capabilities screen to create B2B capabilities for sending a document to the back-end system.

Table 16. Creating B2B capabilities for sending a document

B2B Capabilities column	Document flow definition to enable
Set Target	Enable each document-type component in the back-end document's document flow definition.

Important: If your Community Manager will also be receiving documents from the back-end system, you might want to enable B2B capabilities required while you still have the B2B Capabilities screen displayed. In this case, you enable the component document types of the back-end document's document flow definition to serve as a source. Table 22 on page 33 summarizes the action to take on the B2B Capabilities screen to create B2B capabilities for receiving a document from the back-end system.

Defining the document flow interaction for sending: For the Document Manager to know how to convert the participant document, it must be able to locate a **document flow definition interaction** that combines the document flow definitions for the participant document and the back-end document and identifies which is the source and which is the destination participant.

When the Document Manager is ready to send the converted document to the back-end system, it must be able to locate a participant connection between the source participant and the destination participant (back-end system). However, for a participant connection to exist, a valid document flow definition interaction between the source and the destination documents must exist. To define a document flow definition interaction within Business Integration Connect Enterprise or Advanced edition, click:

Hub Admin > Hub Configuration > Document Flow Definition > Manage Interactions > Create a Valid Interaction

Note: For more information how to create document flow definition interactions in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

To send documents to the back-end system, define an interaction between the source and destination (target) documents, as summarized in Table 17.

Table 17. Creating an interaction for sending a document

Manage Interactions section	Action
Source	Select the component document-types in the <i>participant</i> document's document flow definition.
Target	Select the component document-types in the <i>back-end</i> document's document flow definition.

Note: For more information how to create document flow definition interactions in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

Defining how to connect to the back-end system

For the Document Manager to be able to send the converted document to the back-end system, it must find a valid **participant connection**, which identifies the source and destination participants and provides the location through which these two participants communicate. To create a participant connection, you use the Manage Connections screen in Business Integration Connect. To access this screen, click:

Account Admin > Participant Connections

For a participant connection to be defined, a document flow definition interaction between the source and destination documents must already exist. On the Manage Connections screen, you first check for an existing interaction by specifying the source and destination (target) participants. Table 18 lists the participants to choose on the Manage Connections screen to define a participant connection for sending a document to the back-end system.

Table 18. Creating a participant connection for sending a document

Manage Connection dropdown list	Name of community participant
Source	Name of the community participant that is sending the document to Community Manager
Target	Name of Community Manager, who receives the document from the community participant

Once you specify the Source and Target, you then click Search to check for an existing document flow definition interaction. If no interaction exists, you must create one *before* you can proceed with the creation of a participant connection. If an interaction does exist (one whose source is the participant document flow definition and whose target is the back-end document flow definition), you can configure the participant connection for communication with the back-end system.

Note: For more information how to create a participant connection in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

For back-end integration, this participant connection should specify as its target gateway the gateway you defined in "Defining where to send the participant document" on page 27. You must ensure that all Target Gateways on the Gateways Section screen to the name of the gateway you have created.

Receiving documents from the back-end system

To receive a document from the back-end system, the Community Manager takes the following steps:

1. Receive a document from the back-end system.

The Business Integration Connect Receiver retrieves this source document from a target that has been defined in Community Manager for incoming messages from the back-end system and the associated transfer protocol. When receiving a document from the back-end system, the source document is the document that is received from the *back-end system*; therefore, this document is referred to as the **back-end document**.

2. Convert the back-end document to the destination document, which is in format that the designated community participant requires.

The Document Manager performs this conversion to the destination document. When receiving a document from the back-end system, the destination document is the document that is sent to some *community participant*; therefore, this document is referred to as the **participant document**.

3. Send the participant document to the appropriate community participant.

The Document Manager sends the participant document through a gateway that has been defined in Community Manager for outgoing messages to the appropriate community participant.

Therefore, for the Community Manager to be able to receive a document from the back-end system, you must ensure that the configuration summarized in Table 19 has been performed within Business Integration Connect Enterprise or Advanced edition.

Table 19. Configuration steps to receive documents from the back-end system

Configuration step	Business Integration Connect steps	For more information
1. Define where to retrieve the document.	1. Create a target that receives incoming messages from the back-end system.	“Defining where to retrieve the back-end document”
2. Define how to process the document.	2. Create document flow definitions for the source and destination formats.	“Defining how to process the back-end document” on page 32
	3. Enable B2B capabilities for the document flow definition of the document received from the back-end system.	
	4. Create a document flow definition interaction between the source and destination document flow definitions.	
3. Define how to connect to Business Integration Connect.	5. Create a participant connection that sends documents to Business Integration Connect.	“Defining how to connect to Business Integration Connect” on page 33

Defining where to retrieve the back-end document

To receive documents from the back-end system, the Community Manager must have a target defined. This **target** specifies the source of the documents; that is, it identifies the location (as a URI) at which the Community Manager listens for incoming documents. This location is the same one to which the back-end system sends documents. The target identifies the entrance point into the Receiver (within Business Integration Connect). Within Business Integration Connect, it is the Receiver that checks for a target. Once the Receiver has processed the document, it saves the converted document to the Persistent Shared Storage for later retrieval by the Document Manager.

To define a target within Business Integration Connect Enterprise or Advanced edition, click:

Hub Admin > Hub Configuration > Targets

Note: For more information on how to create a target in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

When you define the target, you specify the transport protocol that the Community Manager and back-end system both use to transfer the back-end document. As Table 20 shows, the choice of transport protocol depends on the format of the document. Its format includes its packaging type and business protocol, which are defined in its document flow definition.

Table 20. Supported transport protocols from back-end system to Business Integration Connect

Packaging type	Business protocol	HTTP or HTTPS?	JMS?	File system?
Backend Integration	RosettaNet (RNSC)	Yes	Yes	No
	XML	Yes	Yes	No
	Binary	Yes	Yes	No
None	XML only	Yes	Yes	Yes
	EDI only	Yes	Yes	Yes
	cXML only	Yes	No	No
	SOAP only	Yes	No	No
	Binary only	No	No	No

Note: The choice of transport protocol also depends on the transport protocols your particular back-end system supports. For more information, refer to the chapter in this guide for integrating to your particular back-end system.

Once you have selected a valid transport protocol for your document, you can provide the other information you need to define for the target in the Target Details screen.

Defining how to process the back-end document

For the Document Manager to be able to process the back-end document, it must know the format to which it needs to convert this document; that is, it needs to know the format of the participant document. As part of the back-end integration, you must ensure that the entities summarized in Table 21 are defined within your Business Integration Connect.

Table 21. Defining how to convert the back-end document

Step	For more information
1. Document flow definitions must exist to define the format of <i>both</i> the participant document and the back-end document.	“Defining the document flow definitions” on page 28
2. The Community Manager’s B2B capabilities must include enablement of the back-end document’s document flow definition as a source.	“Setting B2B capabilities for receiving” on page 33
3. A document flow definition interaction must exist that brings together the back-end document as the source and the participant document as the destination.	“Defining an interaction for receiving” on page 33

Setting B2B capabilities for receiving: For a summary of B2B capabilities as they apply to back-end integration, see “Setting the B2B capabilities for sending” on page 28. This section summarizes how to set B2B capabilities for receiving a document from the back-end system.

For back-end integration, make sure each of the component document types for the back-end document’s document flow definition has been enabled to serve as a source. Table 22 summarizes the action to take on the B2B Capabilities screen to create B2B capabilities for receiving a document from the back-end system.

Table 22. Creating B2B capabilities for receiving a document

B2B Capabilities column	Document flow definition to enable
Set Source	Enable each document-type component in the back-end document’s document flow definition

Defining an interaction for receiving: For a summary of document flow definitions interactions as they apply to back-end integration, see “Defining the document flow interaction for sending” on page 29. This section summarizes how to define the interaction for receiving a document from the back-end system.

To receive documents from the back-end system, define an interaction between the document flow definitions of the source and destination (target) documents as summarized in Table 23.

Table 23. Creating an interaction for receiving a document

Manage Interactions section	Action
Source	Select the component document-types in the <i>back-end</i> document’s document flow definition.
Target	Select the component document-types in the <i>participant</i> document’s document flow definition.

Defining how to connect to Business Integration Connect

For the Receiver to be able to retrieve the document from the back-end system, it must find a valid **participant connection**, which identifies the source and destination participants and provides the location through which these two participants communicate. For a summary of participant connections as they apply to back-end integration, see “Defining how to connect to the back-end system” on page 30.

Table 24 lists the participants to choose on the Manage Connections screen in Business Integration Connect to define a participant connection for receiving a document from the back-end system.

Table 24. Creating a participant connection for receiving a document

Manage Connection dropdown list	Name of community participant
Source	Name of Community Manager, who receives the document from the community participant
Target	Name of the community participant that is sending the document to Community Manager

Once you specify the Source and Target, you then click Search to check for an existing document flow definition interaction. If no interaction exists, you must create one *before* you can proceed with the creation of a participant connection. If an interaction does exist (one whose source is the back-end document flow definition and whose target is the participant document flow definition), you can configure the participant connection for communication with the back-end system.

Note: For more information how to create a participant connection in Business Integration Connect Enterprise or Advanced edition, see the *Hub Configuration Guide*.

For back-end integration, this participant connection should specify as its target gateway the gateway you defined in “Defining where to send the participant document” on page 27. You must ensure that all Target Gateways on the Gateways Section screen to the name of the gateway you have created.

Part 2. Integrating with WebSphere InterChange Server

Chapter 2. Introduction to InterChange Server Integration

This chapter describes how to integrate WebSphere Business Integration Connect with WebSphere InterChange Server.

Notes:

1. For a description of the general process used to integrate Business Integration Connect with a back-end system, see Chapter 1, "Planning for back-end integration," on page 3.
2. This chapter assumes that you are familiar with WebSphere InterChange Server and associated components, such as collaborations, business objects, adapters, and WebSphere InterChange Server Access.

Often integration of WebSphere Business Integration Connect with a back-end system is done by two separate people or roles. Each role configures a particular component, for which that role has expertise. Therefore, this chapter separates the integration with WebSphere InterChange Server into the configuration of Business Integration Connect and the configuration of InterChange Server. Table 25 lists these configuration roles along with the places in this chapter to obtain the associated configuration information.

Table 25. Roles for InterChange Server integration

Configuration role	For more information
Configuration of WebSphere Business Integration Connect	1. "Planning for integration with InterChange Server." 2. "Configuring Business Integration Connect for InterChange Server" on page 41.
Configuration of WebSphere InterChange Server	1. "Planning for integration with InterChange Server." 2. "Configuring InterChange Server" on page 44.

Note: While each of these configuration roles can be performed separately, each also requires common information so that the two components can communicate.

This chapter provides the following information:

- "Planning for integration with InterChange Server"
- "Configuring Business Integration Connect for InterChange Server" on page 41
- "Configuring InterChange Server" on page 44
- "Handling documents with attachments" on page 49

Planning for integration with InterChange Server

To plan for your integration to WebSphere InterChange Server, follow the steps outlined in "Planning the back-end integration" on page 5. Table 26 summarizes the integration steps to integrate WebSphere Business Integration Connect with InterChange Server (ICS).

Table 26. Planning for integration with WebSphere InterChange Server

Integration step	For more information
1. Confirm that you have a supported version of WebSphere InterChange Server installed and available to WebSphere Business Integration Connect.	Chapter 2: "InterChange Server versions that Business Integration Connect supports"
2. Determine the business protocol of the WebSphere Business Integration Connect document.	Chapter 1: "What business protocol are you using?" on page 5
3. Determine the packaging type for the document: None or Backend Integration.	Chapter 1: "Which packaging will you use?" on page 10
4. Determine the transport protocol to use between WebSphere Business Integration Connect and WebSphere InterChange Server.	Chapter 2: "Message transports that InterChange Server supports"
5. Configure WebSphere Business Integration Connect.	Chapter 2: "Configuring Business Integration Connect for InterChange Server" on page 41
6. Configure WebSphere InterChange Server components for use over the chosen transport protocol.	Chapter 2: "Configuring InterChange Server" on page 44

InterChange Server versions that Business Integration Connect supports

Version 4.2.2 of Business Integration Connect can support integration with the following versions of InterChange Server:

- 4.1.1
- 4.2.0
- 4.2.1
- 4.2.2

InterChange Server is available on several platforms, including Windows 2000 and several UNIX-based platforms. For more information, consult your installation guide for InterChange Server in the WebSphere InterChange Server documentation set.

Message transports that InterChange Server supports

When Business Integration Connect sends your message to InterChange Server over a particular message transport protocol, it sends that message to the appropriate **InterChange Server-compatible component**, which understands the particular transport protocol and routes the message to InterChange Server. Similarly when InterChange Server sends a message to Business Integration Connect, it sends the message to the appropriate ICS-compatible component for routing to Business Integration Connect over the appropriate transport protocol.

Table 27 summarizes the ICS-compatible components used in Business Integration Connect integration.

Table 27. InterChange Server-compatible components

ICS-compatible component	Description	Transport protocols
WebSphere Business Integration Adapter	Supports communication between InterChange Server and an application or technology. In this case, the application is Business Integration Connect.	HTTP, JMS
WebSphere Business Integration data handler	Handles the actual conversion from serialized data to business object, or from business object to serialized data. The data handler appropriate for the payload type is used to perform these conversions.	HTTP, JMS
WebSphere Business Integration Connect Servlet	Routes non-SOAP documents to InterChange Server over the HTTP transport protocol	HTTP (documents sent to InterChange Server)

Business Integration Connect supports the message-transport protocols shown in Table 10 on page 18. Of these supported protocols, the following two message-transport protocols are supported by InterChange Server:

- HTTP transport protocol

Note: The exchange of Web Services over HTTP is handled in a separate section because Web Services are exchanged in a manner that is different from other documents transmitted over HTTP. See “Sending SOAP documents over HTTP/S” on page 110.

- JMS transport protocol

Note: InterChange Server provides other types of integration options, such as file-based integration. Refer to the WebSphere InterChange Server documentation for details on enabling the exchange of documents through file-based integration.

Use the transport protocol that best suits the needs of your business. Consider the following:

- First and foremost, determine that the transport protocol you are using between the community participant and Business Integration Connect is available with the integration mechanism used. See “Which message transport will you use?” on page 18.
- Sending SOAP documents to and receiving SOAP documents from the WebSphere InterChange Server requires use of the HTTP transport protocol. For more information, see “Sending SOAP documents over HTTP/S” on page 110.

Using HTTP with InterChange Server

Which ICS-compatible components are required to send and receive documents between Business Integration Connect and InterChange Server over HTTP depends on the following:

- The type of document you are sending
- The version of InterChange Server with which you are integrating.

Note: All references to the HTTP transport protocol apply to HTTPS as well.

Table 28 summarizes where to find information on how to configure ICS-compatible components for use with InterChange Server.

Table 28. Configuring for HTTP transport with InterChange Server

Condition	For more information
If you are transferring non-SOAP documents with InterChange Server whose release version is <i>before version 4.2.2</i>	“Using HTTP transport protocol with v4.2.2 ICS” on page 95
If you are transferring non-SOAP documents with InterChange Server <i>version 4.2.2</i>	“Using HTTP transport protocol with pre-4.2.2 ICS” on page 67
If you are sending SOAP documents	“Sending SOAP documents over HTTP/S” on page 110

As you decide which message transport to use with InterChange Server, consider the following benefit of using HTTP:

- If you require synchronous transactions, you *must* use the HTTP transport protocol.

Using JMS with InterChange Server

The ICS-compatible components required to send and receive documents between Business Integration Connect and InterChange Server over JMS are summarized in Table 71 on page 114.. Basically, support for JMS involves the use of the WebSphere Business Integration Adapter for JMS. The Adapter for JMS invokes collaborations within InterChange Server asynchronously.

As you decide which message transport to use with InterChange Server, consider the following benefit of using JMS:

- The Adapter for JMS can provide “guaranteed event delivery” from Business Integration Connect to the WebSphere InterChange Server.
Guaranteed event delivery ensures that events are never lost or sent twice.

For more information on how to configure for JMS, see Chapter 2, “Introduction to InterChange Server Integration,” on page 37.

Support for InterChange Server integration

Business Integration Connect provides samples to assist you in the integration process with InterChange Server. These samples reside in the following subdirectory of the Business Integration Connect product directory:

Integration/WBI/WICS/samples

Table 29 lists the subdirectories of the samples directory for the different transport protocols that InterChange Server supports.

Table 29. Samples for InterChange Server integration

Transport protocol	InterChange Server version	Samples subdirectory
HTTP	4.1.1, 4.2.0, 4.2.1	WBICServlet
HTTP	4.2.2	General samples: HTTP
JMS	All supported versions	RosettaNet-specific samples: RosettaNet/HTTP General samples: JMS RosettaNet-specific samples: RosettaNet/JMS

Configuring Business Integration Connect for InterChange Server

A general overview of how to configure Business Integration Connect to communicate with a back-end system is provided in “Configuring Business Integration Connect” on page 25. This section summarizes the steps needed to configure Business Integration Connect to communicate with InterChange Server. To perform this configuration, use an instance of Business Integration Connect Enterprise or Advanced Edition that functions as the Community Manager in your hub community.

Configuration of Business Integration Connect involves the following steps:

- Configuring for support of outgoing documents
For information on sending documents from Business Integration Connect to InterChange Server, see “Providing support for outgoing documents.”
- Configuring for incoming documents
For information on sending documents from InterChange Server to Business Integration Connect, see “Providing support for incoming documents” on page 42.

Providing support for outgoing documents

For Business Integration Connect to send documents to any back-end system, you must perform the steps described in “Defining where to send the participant document” on page 27. When your back-end system is InterChange Server, you need to create a gateway whose transport type matches the transport protocol used for messages between Business Integration Connect and InterChange Server. When the Community Manager sends a document to InterChange Server, it must know where to route the document. This location must conform with the transport protocol being used. The transport protocol must be one that InterChange Server supports (see “Message transports that InterChange Server supports” on page 38).

The following sections summarize how to create gateways for following transport protocols, which InterChange Server supports:

- “Configuring for outgoing documents over HTTP transport protocol”
- “Configuring for outgoing documents over JMS transport protocol” on page 42

Configuring for outgoing documents over HTTP transport protocol

When the Community Manager sends a document to InterChange Server over the HTTP protocol, the Community Manager routes the message through the defined gateway. This gateway identifies the URL where the document can be received by InterChange Server. When InterChange Server uses the HTTP protocol, an ICS-compatible component receives the document at the appropriate URL, where it can then send it to InterChange Server.

For the Community Manager to be able to send documents through a gateway over the HTTP transport protocol, you must create a gateway from the Gateway Details screen of the Community Console. This gateway must be configured to use the HTTP 1.1. transport protocol and to write to the URL on which the appropriate ICS-compatible component is listening. As Table 30 shows, you provide this URL in the Target URI field of the gateway definition.

Note: An overview of how to create a gateway is provided in “Defining where to send the participant document” on page 27.

Table 30. HTTP values for Gateway Details screen for communication with InterChange Server

InterChange Server version	Value of Target URI field	Notes and restrictions
4.1.1, 4.2.0, 4.2.1	The URL should be the same as the one configured for the WebSphere Business Integration Connect Servlet.	Obtain this URL from the configuration of the Connect Servlet in the WebSphere InterChange Server integration.
4.2.2	The URL should be the same as the the one that the protocol listener of the WebSphere Business Integration Adapter for HTTP uses to receive requests.	Obtain this URL from the configuration of the Adapter for HTTP in the WebSphere InterChange Server integration.

Configuring for outgoing documents over JMS transport protocol

When the Community Manager sends documents to InterChange Server over the JMS protocol, the Community Manager routes the document to the appropriate JMS queue, where it can be retrieved by InterChange Server. For the Community Manager to obtain this JMS location, you must create a gateway in Business Integration Connect, one that uses the JMS transport protocol. This gateway must be configured to write to the queue on which the Adapter for JMS polls.

Note: For an overview of how to create a gateway, see “Defining where to send the participant document” on page 27.

For the Community Manager to be able to send documents through a gateway over the JMS transport protocol, create a gateway from the Gateway Details screen of the Community Console. If you are using WebSphere MQ version 5.3 as your JMS provider, use the information in Table 116 on page 179 to set the gateway fields. In addition, specify the information specified in Table 31 for the JMS protocol in the Gateway Details screen.

Table 31. JMS values for the Gateway Details screen for communication with ICS

Gateway Details field	Value	Notes and restrictions
JMS Message Class	TextMessage, BytesMessage, or StreamMessage	Versions of the Adapter for JMS <i>before</i> 2.4.1 supports JMS text messages only. If you are using a pre-2.4.1 version of this adapter, make sure that the gateway is configured to write <i>only</i> JMS text messages (TextMessage)
JMS Queue Name	Same JMS queue name as the input queue for the Adapter for JMS	This queue must be included the in the list of input queues of the Adapter for JMS; that is, the adapter must poll this queue for incoming events. For more information, see “Identifying the JMS queues” on page 121..

Providing support for incoming documents

For Business Integration Connect to receive messages from any back-end system, you must perform the steps described in “Defining where to retrieve the back-end

document” on page 31. When your back-end system is InterChange Server, you need to take the following steps in your Community Manager:

1. As part of your participant profile for the Community Manager, define the gateway type and provide the associated IP address on which the Receiver will listen.
2. Create a target whose transport type matches the transport protocol used for documents between Business Integration Connect and InterChange Server.
For Community Manager to receive a document from InterChange Server, it must know the location at which to retrieve the messages. This location must conform with the transport protocol to be used. The transport protocol must be one that InterChange Server supports (see “Message transports that InterChange Server supports” on page 38).

The following sections summarize how to create targets for transport protocols that InterChange Server supports.

Configuring for incoming documents over HTTP transport protocol

When the Community Manager receives a document over the HTTP transport protocol, its Receiver retrieves the document from the defined target. This target identifies the URL at which the Receiver listens for documents from InterChange Server. When InterChange Server uses the HTTP transport protocol, an adapter sends the document to the appropriate URL, where it can be received by the Community Manager.

For the Community Manager to receive documents through a target over the HTTP transport protocol, you must create a target from the Target List screen of the Community Console. This target must use the HTTP 1.1 transport protocol. The Community Manager determines this URL as a combination of the following information:

- The IP address of the host machine, obtained from within the Community Manager’s participant profile
- The target URL, obtained from the URL field of the target definition

Note: An overview of how to create a target is provided in “Defining where to retrieve the back-end document” on page 31.

For InterChange Server to be able to send documents to this target, its ICS-compatible component must be configured to send documents to this URL. Therefore, you must ensure that this target URL is available for the InterChange Server configuration.

Configuring for incoming documents over JMS transport protocol

When the Community Manager receives documents from InterChange Server over the JMS protocol, the Community Manager obtains the document from the appropriate JMS queue, where InterChange Server has sent it. For the Community Manager to be able to obtain this JMS location, you must create a target in Business Integration Connect, one that uses the JMS transport protocol. Through the target, the Community Manager listens for any documents on its input queue and retrieves them.

Note: For an overview of how to create a target, see “Defining where to retrieve the back-end document” on page 31.

For the Community Manager to receive documents through a target over the JMS transport, you must create a target from the Target List screen of the Community Console. If you are using WebSphere MQ version 5.3 as your JMS provider, use the information in Table 115 on page 178 to set the target fields. In addition, specify the information specified in Table 32 for the JMS protocol in the Target Details screen.

Table 32. JMS values for the Target Details screen for communication with ICS

Target Details field	Value	Notes and restrictions
JMS Message Class	TextMessage	Versions of the Adapter for JMS <i>before</i> 2.4.1 support JMS text messages <i>only</i> . If you are using one of these versions of the adapter, make sure that the target is configured to write <i>only</i> JMS text messages.
JMS Queue Name	Same JMS queue name as the output queue for the Adapter for JMS	This queue must be listed as the output queue of the Adapter for JMS; that is, the adapter must send documents to this queue. For more information, see “Identifying the JMS queues” on page 121.

Configuring InterChange Server

For your interactions between Business Integration Connect and InterChange Server, you must create an Integration Component Library (ICL) within the System Manager tool. This ICL will include the following artifacts:

- Business object definitions
- Connector objects
- Collaboration templates and collaboration objects

You must also create a user product and select from the ICL those artifacts required for your particular interaction between InterChange Server and Business Integration Connect.

Note: For more information on how to create ICLs and configure InterChange Server, see the *System Implementation Guide* in the WebSphere InterChange Server documentation set.

Creating business object definitions

Business Integration Connect sends your message to an ICS-compatible component, which routes the message to InterChange Server in the form of one or more **business objects**. For InterChange Server to recognize a business object, it must first locate a template, called a **business object definition**, to describe the structure of the information in the business object. Each piece of information in a business object definition is held in an **attribute**. Therefore, you must create business object definitions to represent the information in your message. To create business object definitions, use the Business Object Designer tool to create business object definitions.

Note: Business Object Designer is included as part of both the WebSphere InterChange Server and the WebSphere Business Integration Adapter products. For more information on the use of this tool, see the *Business Object Development Guide*.

InterChange Server uses business objects for the following information:

- “Business object for the document”
- “Business objects for configuration information” on page 47

Business object for the document

To hold the payload of the Business Integration Connect document or message, you must define a business object definition to represent the **payload business object**. It is in the form of a payload business object that the ICS-compatible component transfers the document into (or out of) InterChange Server. This section provides the following information on the payload business object:

- “Business object structure”
- “Business object conversion” on page 46
- “InterChange Server terminology” on page 47

Business object structure: The payload business object must be designed so that each piece of information in the document that you want to transfer must have an attribute in its associated payload business object definition. As Table 33 shows, the contents of the payload business object depends on the structure of the document and the packaging type that the document uses.

Table 33. Relationship of packaging to the structure of the payload business object

Document structure	Packaging type	Payload business object definition
Payload only	None	Holds the payload information of the document.
Payload only	Backend Integration	Holds: <ul style="list-style-type: none"> • The payload information of the document • Transport-level headers
Payload and attachments	None	<i>Not applicable.</i> You must use Backend Integration packaging if your document contains attachments.
Payload and attachments	Backend Integration	Holds: <ul style="list-style-type: none"> • The payload information of the document • Transport-level headers • The attachment container, which holds the attachment data and any attachment business objects <p>A Business Integration Connect-supplied data handler, called the Attachment data handler, is required to process the transport envelope. For more information, see “Handling documents with attachments” on page 49.</p>
The document contains an XML wrapper, called a transport envelope, in which both the payload and attachments are wrapped.		

The payload business object must also be designed in according to the requirements of the particular ICS-compatible component used for integration with Business Integration Connect (see Table 27 on page 39). Table 34 provides information on where to find details of how to create the payload business object for transfer over a particular transport protocol.

Table 34. Creating payload business objects for different transport protocols

Transport protocol	Notes and restrictions	For more information
HTTP	Use for a pre-4.2.2 InterChange Server	"Creating business object definitions for pre-4.2.2 ICS over HTTP" on page 87
HTTP	Use for a version 4.2.2 InterChange Server	"Creating business object definitions for v4.2.2 ICS over HTTP" on page 102
JMS	If document uses Backend Integration packaging	"Creating business object definitions for JMS" on page 121
All	If document has attachments	"Creating attachment-related business object definitions" on page 60

Business object conversion: Usually, the ICS-compatible component uses a data handler to convert between the format of the document and its business-object representation. This data handler is called the **payload data handler**. The ICS-compatible component must be configured to call appropriate data handler for the payload's content type. Usually, the WebSphere Business Integration Data Handler for XML is configured as the payload data handler because it converts between XML messages and business objects. However, you can create custom data handlers for any message formats that do not have a corresponding data handler provided by WebSphere Business Integration Server.

Note: For processing XML messages, make sure you are using the WebSphere Business Integration Data Handler for XML Version 2.3.1 or higher. For cXML messages, you must use the Data Handler for XML Version 2.4.1 or higher.

You must make sure the payload data handlers you are using can ignore the child meta-object that are required by the transport protocol you are using. Before using a data handler (whether it is supplied by WebSphere Business Integration or whether it is a custom data handler), make sure it provides support for child meta-objects. Refer to the section on the `cw_mo_label` tag in the business object's application-specific information in the appropriate section for your transport protocol (see Table 34).

To indicate which data handler to use to convert the payload, you must take the following steps:

- Identify the MIME type that the data handler must support to convert the payload and locate a data handler that can handle this MIME type.
The *Data Handler Guide* in the WebSphere Business Integration Adapter documentation set describes the data handlers that IBM provides. If none of these data handlers can work, you can create a custom data handler.
- In Business Object Designer, create a child meta-object for the data handler you need to use. If you are using an IBM-provided data handler, refer to the *Data Handler Guide* for information on the structure of the child meta-object.
- In Business Object Designer, update the top-level data-handler meta-object for connectors to include an attribute for the supported MIME type. The attribute type for this attribute is the data handler's child meta-object.
- In Connector Configurator, set the appropriate connector configuration properties to identify the data handler to use:

- Set the `DataHandlerConfigMO` and `DataHandlerMimeType` properties with the name of the top-level data-handler meta-object and the supported MIME type, respectively.
- Set the `DataHandlerClassName` property with the name of the data-handler class to instantiate.

Note: You set *either* the `DataHandlerConfigMO` and `DataHandlerMimeType` properties *or* the `DataHandlerClassName` property.

- In Connector Configurator, include the top-level data-handler meta-object in the list of supported business objects.

InterChange Server terminology: For InterChange Server, the name of the payload business object depends on the direction of the communication, as follows:

- When Business Integration Connect *sends* a document to InterChange Server, it is participating in InterChange Server’s **event notification**.

In this case, the data business object is called the **event business object** (sometimes called just an event), which notifies InterChange Server of an event that occurred in some community participant.

- When Business Integration Connect *receives* a document from InterChange Server, it is participating in InterChange Server’s **request processing**.

In this case, the data business object is a **request business object**, which InterChange Server has sent to request information from some community participant. In response, InterChange Server might send a **response business object** back to the hub community.

Business objects for configuration information

For many of the ICS-compatible components, you create business object definitions to hold configuration information. Such business objects are often called **meta-objects**.

Table 34 provides information on where to find details of how to create the data business object for transfer over a particular transport protocol.

Table 35.

Transport protocol	Related component	For more information
HTTP (to pre-4.2.2 InterChange Server)	Wrapper data handler	“Creating the configuration business objects for the Wrapper data handler” on page 77
HTTP (to v4.2.2 InterChange Server)	Adapter for HTTP	“Creating HTTP transport-level header information for pre-4.2.2 InterChange Server” on page 92.
JMS	Adapter for JMS	“Creating JMS header information” on page 122
All	Attachment data handler	“Creating the Attachment child meta-object” on page 56

Creating the connectors

If the ICS-compatible component for your transport protocol is a WebSphere Business Integration Adapter, you must create a **connector object** for that adapter.

This connector object represents an instance of the adapter at run-time. You create connector objects within InterChange Server's System Manager tool.

Note: For information on how to create connector objects, refer to the *System Implementation Guide* in the WebSphere InterChange Server documentation set.

Table 36 summarizes where to find information about how to create connector objects, based on the transport protocol you are using.

Table 36. Creating connector objects for different transport protocols

Transport protocol	ICS-compatible component	For more information
HTTP (with a pre-4.2.2 InterChange Server)	Adapter for XML (Request processing only)	"Creating the XML connector object" on page 95
HTTP (with a version 4.2.2 InterChange Server)	Adapter for HTTP	"Creating the HTTP connector object" on page 109
JMS	Adapter for JMS	"Creating the JMS connector object" on page 126

Creating the collaborations

It is the **collaboration**, within InterChange Server, that performs the actual business process you need. Therefore, the appropriate collaboration must exist for InterChange Server to correctly process your Business Integration Connect documents. Make sure you take the following steps to make the appropriate collaboration available at run-time:

1. Ensure that a collaboration template exists that provides the business process you need:
 - If such a collaboration template does *not* currently exist, you must create one and compile it.
 - If a collaboration template does exist, you must understand how to use it sufficiently to be able to configure its collaboration object.
2. Create a collaboration object and bind its ports, as follows:
 - For request processing: the "to" port, which sends requests to Business Integration Connect, should be set to the ICS-compatible component.
 - For event notification: the "from" port, which receives events from Business Integration Connect, should be set to the ICS-compatible component.

Table 37 summarizes where to find information about how to create connector objects, based on the transport protocol you are using.

Table 37. Collaboration binding for different transport protocols

Transport protocol	ICS-compatible component	For more information
HTTP (with a pre-4.2.2 InterChange Server)	Adapter for XML (Request processing only)	"Binding collaborations to communicate with Adapter for XML" on page 95

Table 37. Collaboration binding for different transport protocols (continued)

Transport protocol	ICS-compatible component	For more information
HTTP (with a version 4.2.2 InterChange Server)	Adapter for HTTP	“Binding collaborations to communicate with Adapter for HTTP” on page 110
JMS	Adapter for JMS	“Binding collaborations to communicate with Adapter for JMS” on page 126

Deploy the project

Once your user project contains the artifacts that define the run-time components needed, you must deploy it to the InterChange Server repository. You deploy a user project within System Manager.

Handling documents with attachments

Business Integration Connect provides the Attachment data handler to process documents that are sent between Business Integration Connect to InterChange Server and InterChange Server. The Attachment data handler converts a document within the XML **transport envelope** (with or without attachments) between its serialized format and its business-object representation. You should configure the Attachment data handler as the payload data handler in either of the following cases:

- The Envelope Flag for Backend Integration packaging has been set to Yes.
When this flag is set to Yes, Business Integration Connect always wraps a document in an XML transport envelope, regardless of whether it contains attachments. You set this flag to Yes for Backend Integration packaging as part of the profile’s B2B Capabilities screen. For more information, see “Payload” on page 16.
- The document to be processed can contain attachments.
When a document contains attachments, Business Integration Connect wraps it in an XML transport envelope. In any document flow, there is one payload and, optionally, multiple attachments. If you are sending or receiving documents that contain attachments, your payload business object needs to contain attachment information.

Note: The Attachment data handler is *not* required for SOAP documents that contain attachments. For information on how SOAP documents are handled, see “Sending SOAP documents over HTTP/S” on page 110.

The Attachment data handler can be called in any of the following contexts:

- From a WebSphere Business Integration Adapter
 - If Business Integration Connect and a *version 4.2.2* InterChange Server use the HTTP transport protocol, it is the Adapter for HTTP that calls the Attachment data handler.
 - If Business Integration Connect and a *pre-4.2.2* version InterChange Server participate in *request processing* (InterChange Server initiates the request), it is the Adapter for XML that calls the Attachment data handler to convert business objects into a serialized form of the document wrapped in an XML transport envelope.

- If Business Integration Connect and InterChange Server (of *any* supported version) use the JMS transport protocol, it is the Adapter for JMS that calls the Attachment data handler.
- From WebSphere Server Access, which resides within InterChange Server to handle access-client requests

If Business Integration Connect and a *pre-4.2.2 version* InterChange Server participate in *event notification* (Business Integration Connect initiates the request), the WebSphere Business Integration Connect Servlet (an access client) sends the document to Server Access, which calls the Wrapper data handler to convert the document to its business-object representation. If this document is wrapped in an XML transport envelope, the Wrapper data handler calls the Attachment data handler to convert the serialized form of the document (in its XML transport envelope) into the corresponding business object.

Regardless of whether the calling entity is an adapter or Server Access, when the calling entity receives a document within an XML transport envelope, it calls the Attachment data handler to convert this document to its appropriate business-object representation. For example, Figure 20 on page 115 shows the Adapter for JMS calling the Attachment data handler to convert the serialized format of the document to its business-object representation. Conversely, when the calling entity receives a business-object representation for a document within an XML transport envelope, it calls the Attachment data handler to convert this business-object structure to its appropriate document format. For example, Figure 21 on page 117 shows the Adapter for JMS calling the Attachment data handler to convert the business-object representation of the document to its serialized format.

This section provides the following information on the Attachment data handler:

- “Performing the conversion”
- “Setting up the environment for the Attachment data handler” on page 55
- “Configuring the Attachment data handler” on page 56
- “Creating attachment-related business object definitions” on page 60

Performing the conversion

The Attachment data handler can interpret the structure of this XML transport envelope and handle the conversion between the contained data and the corresponding business-object representation, as follows:

- “Converting documents to business objects” to send a document to InterChange Server
- “Converting business objects to documents” on page 53 to receive a document from InterChange Server

Converting documents to business objects

Before Business Integration Connect sends a document to InterChange Server, it must determine whether to wrap the contents in an XML transport envelope. If Business Integration Connect creates the transport envelope, the payload and any attachments are Base64 encoded. Business Integration Connect then sends the XML transport envelope to the appropriate ICS-compatible component with the appropriate transport-level headers. This ICS-compatible component (a WebSphere Business Integration adapter or the Wrapper data handler) can be configured to call the Attachment data handler to handle the conversion of payload and any attachments in an XML-wrapped document into the corresponding business-object representation.

To convert a document wrapped in an XML transport envelope to its business-object representation, the calling entity instantiates the Attachment data handler, passing it the document (in its transport envelope). The Attachment data handler then takes the following steps:

1. Load the content-type maps defined in the data handler's child meta-object.
The content-type maps are defined in the `ContentTypeMap_x` configuration properties of the child meta-object. The child meta-object is a business object that contains configuration information for the Attachment data handler. Attributes in this business object associate content-type maps with content types. For more information, see "Creating the Attachment child meta-object" on page 56.
2. Check the document to see whether it is wrapped in an XML transport envelope.
 - If the Attachment data handler does *not* detect the transport envelope, it does not need to extract the payload from this envelope structure.
The document contains only a payload, which the Attachment data handler must convert to its associated business-object representation. For more information, see "Processing a document not in a transport envelope."
 - If the Attachment data handler *does* detect the transport envelope, it must extract payload and any attachments from this envelope structure.
The document contains a payload and possibly some attachments. Therefore, the Attachment data handler needs to convert the payload *and* any attachments to their associated business-object representation. For more information, see "Processing a document in a transport envelope."
3. Set the resulting payload business object and return this business object to the calling entity.

Processing a document not in a transport envelope: If the Attachment data handler determines that the document is *not* contained in an XML transport envelope, it does not need to extract the payload data from the envelope structure. Therefore, the data handler uses the `PayloadDataHandlerMimeType` configuration property (defined in its child meta-object) to obtain the MIME type that identifies the default payload data handler to instantiate for the document payload. This data handler converts the payload data to its corresponding payload business object and returns the resulting payload business object to its calling entity.

Processing a document in a transport envelope: If the Attachment data handler determines that the document is contained in an XML transport envelope, it must extract the payload and any attachments from this envelope structure before it can process them. Therefore, the data handler takes the following steps to process and convert the document:

1. Extract the payload and any attachments from the transport envelope and decode the payload data.
The payload is contained in the `<payload>` XML tag. Each attachment is contained in an `<attachment>` XML tag.
2. Search the content-type maps for a content type that matches that of the payload.
Use the MIME type specified in the matching content-type map to create an instance of a data handler. This data handler converts the payload data to its corresponding payload business object and returns the resulting business object to the Attachment data handler.
3. Create the content-information business object for the payload.

Examine the business-object-level application-specific information of the payload business object definition and determine the name of the content-information business object, whose attribute name is specified by the `cw_mo_bcg_content_info` tag. Create an instance of this content-information business object and set the values for payload's content type and encoding.

4. Create the attachment-container business object for the payload.

Examine the business-object-level application-specific information of the payload business object and determine the name of the attachment-container business object, whose attribute name is specified by the `cw_mo_bcg_attachment` tag. Create an instance of the attachment-container business object and save it in the appropriate attribute of the payload business object.

If the `cw_mo_bcg_attachment` tag does not exist (or is empty), assume that the document does not contain any attachments. Therefore, no further processing steps are required. The Attachment data handler returns the converted payload business object.

5. Create the default attachment business object for the attachment container.

Examine the business-object-level application-specific information of the attachment-container business object and determine the name of the default attachment business object, whose attribute name is specified by the `cw_mo_bcg_default_attribute` tag. Create an instance of the default attachment business object and save it in the appropriate attribute of the attachment-container business object.

6. Determine whether the attachment needs to be converted to a business object by searching the content-type maps for a content type that matches that of the attachment.

Get the content type and character-set encoding from the attachment and check to see whether there is a corresponding entry in a content-type map.

- If a corresponding content-type map is *not* found, the Attachment data handler does not create a business object for the attachment data.

Therefore, the data handler creates an instance of the default attachment business object, sets the values for the content type and encoding within its content-information business object, and sets the base64-encoded attachment data (as a string) in the attachment attribute.

The Attachment data handler then populates the attachment-container business object with the default attachment business object.

- If a content-type map *is* found, the Attachment data handler checks to see whether the attachment needs to be converted to a business object:
 - If the `ConvertAttachment` configuration property in the matching content-type map is `false`, the Attachment data handler creates an instance of the default attachment business object, sets the values for the content type and encoding within its content-information business object, and sets the base64-encoded attachment data (as a string) in the attachment attribute.
 - If the `ConvertAttachment` configuration property in the matching content-type map is `true`, the Attachment data handler decodes the attachment data and creates an instance of a data handler to process the attachment data. This data handler processes the decoded bytes and returns the corresponding attachment business object.

The Attachment data handler then examines the business-object-level application-specific information of the attachment business object

definition and determines the name of the content-information business object, whose attribute name is specified by the `cw_mo_bcg_content_info` tag. If this tag exists, the data handler creates the content-information business object for the attachment and sets the values for attachment's content type and encoding.

Finally, the Attachment data handler populates the attachment-container business object with the attachment business object.

Converting business objects to documents

Before Business Integration Connect receives a document from InterChange Server, an ICS-compatible component must determine whether to wrap the business-object representation of the payload and any attachments in the XML transport envelope. InterChange Server sends the business object to the appropriate ICS-compatible component, which handles the actual conversion. This ICS-compatible component (a WebSphere Business Integration adapter or the Wrapper data handler) can be configured to call the Attachment data handler to handle the conversion of payload and any attachment business objects into the corresponding payload and attachments as well as the creation of an XML transport envelope.

To convert a payload business object with attachments to its transport-envelope representation, the calling entity instantiates the Attachment data handler, passing it the payload business object. The Attachment data handler takes the following steps:

1. Load the content-type maps defined in its configuration meta-object.

The content-type maps are defined in the `ContentTypeMap_x` configuration properties of the child meta-object. The child meta-object is a business object that contains configuration information for the Attachment data handler. Attributes in this business object associate content-type maps with content types. For more information, see "Creating the Attachment child meta-object" on page 56.

2. Check the business object to determine whether to create an XML transport envelope.

- If the Attachment data handler does *not* determine that the document requires a transport envelope, it does not need to wrap the payload in this envelope structure.

The document contains only a payload, which the Attachment data handler must create from its associated business-object representation. The data handler does not need to create a transport envelope for the document. For more information, see "Creating a document without a transport envelope."

- If the Attachment data handler does determine that the document requires a transport envelope, it must wrap the payload and any attachments in this envelope structure.

The document contains a payload and possibly some attachments. Therefore, the Attachment data handler needs to convert the payload business-object representation to a payload *and* any attachments and wrap these components in a transport envelope. For more information, see "Creating a document with a transport envelope" on page 54.

3. Set the resulting payload and any attachment tags in the Business Integration Connect document and return this document to the calling entity.

Creating a document without a transport envelope: If the Attachment data handler determines that the payload business object does *not* require an XML transport envelope, it does not need to wrap the payload data in the envelope structure. Therefore, the data handler uses the default payload data handler to

convert the payload business object to its corresponding payload document. The `PayloadDataHandlerMimeType` configuration property (defined in the Attachment data handler's child meta-object) contains the MIME type that identifies the default payload data handler to instantiate for the payload business object. This data handler receives the payload business object as an argument and returns the resulting payload document to its calling entity.

Creating a document with a transport envelope: If the Attachment data handler determines that the payload business object *does* require an XML transport envelope, it must wrap the payload and attachment documents in this envelope structure. Therefore, the data handler takes the following steps to process and convert the business object:

1. Get the content type and character-set encoding for the payload.

The `cw_mo_bcg_content_info` tag in the business-object-level application-specific information of the payload business object specifies the name of the content-information attribute. This attribute contains the content type and encoding for the payload.

Note: If the content-information attribute does not exist, use the default data handler (identified by the MIME type contained in the `PayloadDataHandlerMimeType` configuration property, in the Attachment data handler's child meta-object) to convert the payload business object.

2. Search the content-type maps for a content type that matches that of the payload.

Use the MIME type specified in the matching content-type map to create an instance of a payload data handler. This data handler converts the payload business object to its corresponding payload document and returns the resulting document to the Attachment data handler. From the string that is returned by the payload data handler, the Attachment data handler encodes the bytes using Base64 and stores the result in the payload tag of the XML transport envelope.

3. Get the attachment container from the payload business object.

The attachment container resides in the attachment-container attribute of the payload business object. The business-object-level application-specific information of the payload business object contains the `cw_mo_bcg_attachment` tag, which identifies the attachment-container attribute. This attribute contains the attachments.

If the `cw_mo_bcg_attachment` tag does not exist (or is empty), assume that the document does not contain any attachments. Therefore, no further processing steps are required. The Attachment data handler returns the converted payload in its transport envelope.

4. For each attachment, determine whether the attachment is represented as a business object or just data.

- If the attachment is just attachment data, the business-object-level application-specific information of the attachment-container business object contains the `cw_mo_bcg_default_attribute` tag, which identifies the default-attachment attribute. This attribute contains the attachment data, which the Attachment data handler retrieves, extracts the Base64-encoded data, and stores the result in the document.
- If the attachment is represented by a business object, its attribute-level application-specific information contains the `wbic_type` tag to indicate that it contains an attachment business object.

The Attachment data handler takes the following steps to process the attachment business object:

- a. Retrieve the contents of the attachment attribute and get the content type and encoding for the attachment.

The business-object-level application-specific information of the attachment business object contains the `cw_mo_bcg_content_info` tag, which identifies the content-information attribute. This attribute contains the content type and encoding for this attachment. The Attachment data handler stores this content information in the attachment tag of the document.

- b. Search the content-type maps for a content type that matches that of the attachment.

Use the MIME type in the matching content-type map to create an instance of a data handler. This data handler converts the attachment business object to its corresponding attachment document and returns the resulting document (as a string) to the Attachment data handler.

- c. Store the encoded result in the attachment tag of the XML wrapper for the document.

The Attachment data handler gets the bytes from the returned string (using the character set, if one was present) and encodes the bytes using Base64. It then stores the result in the attachment tag.

Setting up the environment for the Attachment data handler

Use of the Business Integration Connect-supplied Attachment data handler involves the following steps:

- “Deploying the Attachment data handler”
- “Configuring the Attachment data handler” on page 56

Deploying the Attachment data handler

The Attachment data handler and associated repository file are available on the Business Integration Connect installation medium, in the locations listed in Table 38.

Table 38. Location of the components for Attachment data handler

Component	Location
Attachment data handler	Integration/WBI/WICS/Attachment/ bcgwiattachmentdh.jar
Repository file	Integration/WBI/WICS/Attachment/ MO_DataHandler_DefaultAttachmentConfig.in

Deploy the files into the Web server according to the documentation for the Web server.

Specifying the location of the Attachment data handler

WebSphere InterChange Server needs to know the location of the Attachment data handler, so that it can load it at run-time. To specify the location of the Attachment data handler, take the following steps:

1. Edit the ICS startup script, `start_server.bat`, which is located in the `bin` subdirectory of the InterChange Server product directory (on the machine where InterChange Server resides).
2. To the `CLASSPATH` variable in this file, add the jar file for the Attachment data handler:

Add the jar file for the Attachment data handler, `bcgwbiattachmentdh.jar`, to the list of jar files included at ICS startup.

Configuring the Attachment data handler

Configuring the Attachment data handler consists of the following steps to create the configuration business objects:

- “Creating the Attachment child meta-object”
- “Updating the top-level data-handler meta-object” on page 58

Note: You must also create the attachment-related business object definitions for the Attachment data handler. For more information, see “Creating attachment-related business object definitions” on page 60.

Creating the Attachment child meta-object

To configure the Attachment data handler, you must create a child meta-object to provide the class name and configuration properties that the Attachment data handler needs. To create this meta-object, you create a business object definition that contains the attributes listed in Table 39. Use Business Object Designer, which is part of the WebSphere Business Integration Toolset, to create this business object definition

The child meta-object provides the class name and configuration properties that the Attachment data handler needs. In the Business Object Designer tool, create a child meta-object that includes MIME types for the payload and for the types of attachments you expect to receive.

The attributes of the child meta-object are shown in Table 39. An example of a child meta-object for the Attachment data handler is shown in Figure 6 on page 58.

Note: The sample business objects shown in this chapter do *not* include the standard attributes (such as `ObjectEventId`) required by WebSphere InterChange Server but not used by the Attachment data handler.

Table 39. Configuration properties in the Attachment child meta-object

Attribute Name	Description
<code>ClassName</code>	Class name (required), which points to the following data handler class: <code>com.ibm.bcg.DataHandlers.AttachmentDataHandler</code>
<code>ContentTypeMap_x</code>	The content-type map for the payload and for each type of attachment you expect to receive in the XML wrapper.
<code>PayloadDataHandlerMimeType</code>	For more information, see “Content-type maps.” MIME type used to identify the default data handler, which processes a payload that does <i>not</i> have associated attachments.

Important: To assign a value to the attributes in Table 39., set the default value of the attribute. For example, if the Attachment data handler is to use the XML data handler for its default data handler, set the Default Value of the `PayloadDataHandlerMimeType` attribute to `text/xml`.

Content-type maps: The **content-type map** determines the data handler that the Attachment data handler calls to convert information formatted in the associated content type. For example, if the content type of the payload is `application/xml`,

the Attachment data handler looks for a content-type map whose `ContentType` attribute contains the value `application/xml`. If no matching content type can be found, the data handler assumes that it should *not* convert the associated attachment to a business object.

You would create a content-type map for each of these content types, with the attribute-level application-specific information as shown in Table 41.

When you create an attribute in the child meta-object that represents a content-type map, keep the following in mind:

- The name of the content-type-map attribute has the following format:
`ContentTypeMap_x`
 where *x* is an integer that uniquely identifies the content-type map within the business object definition.

Note: You must order the `ContentTypeMap_x` attributes in sequence. For example, if you have three content-type maps, their attributes must be named `ContentType_1`, `ContentType_2`, and `ContentType_3`.

- The default value of the content-type-map attribute must contain some combination of valid tags.

Table 40 lists the tags that the default value for this attribute can contain.

Table 40. Valid tags for default value of content-type-map attribute

Tag name	Description	Required?
<code>ContentType</code>	Actual content type that comes in the transport envelope (for example, <code>text/xml</code>).	Yes
<code>MimeType</code>	MIME type used to identify the data handler to convert the associated content type to a business object. If you do not specify <code>MimeType</code> , the data handler uses the value of <code>ContentType</code> to instantiate the data handler.	No
<code>CharSet</code>	Name of a character set (for example, UTF-8) that the Attachment data handler uses to convert bytes to a string or a string to bytes. If you do not specify <code>CharSet</code> , the Attachment data handler takes the following actions: <ul style="list-style-type: none"> • For inbound data, the data bytes that result from decoding the message from base64 are used for the conversion to the business object. • For outbound data, calls are made to the method of the child data handler that returns bytes (and not a string). 	No
<code>ConvertAttachment</code>	Boolean value to indicate whether the attachment should be converted to a business object. The default is <code>false</code> .	No

The content-type map can also specify the character set for encoding as well as whether an attachment should be converted to a business object. For a description of the child meta-object attributes and an example, see “Creating the Attachment child meta-object” on page 56.

For example, suppose you have the following content types in your document:

- `application/xml`

- text/xml
- application/octet-stream

Table 41. Sample content-type maps

Content type	Attribute name	Default value
text/xml	ContentType_1	ContentType=text/xml;MimeType=myxml; CharSet=UTF-8;ConvertAttachment=false;
application/xml	ContentType_2	ContentType=application/xml; MimeType=mynewxml;CharSet=UTF-16; ConvertAttachment=true;
application/octet-stream	ContentType_3	ContentType=application/octet-stream; MimeType=myoctet

Sample child meta-object: Business Integration Connect provides the following InterChange Server repository input file, which contains a sample child meta-object for the Attachment data handler:

```
ProductDir/Integration/WBI/WICS/Attachment/
MO_DataHandler_DefaultAttachmentConfig.in
```

where *ProductDir* is the directory of your installed Business Integration Connect product. This repository file defines a single Attachment data handler whose MIME type is *wbic_attachment* and whose associated child meta-object is *MO_DataHandler_DefaultAttachmentConfig*. Figure 6 shows the sample child meta-object for the Attachment data handler. This meta-object defines two content-type maps, *ContentTypeMap_1* and *ContentTypeMap_2*.

MO_DataHandler_DefaultAttachmentConfig
Name = ClassName Default Value = com.ibm.bcg.DataHandlers. AttachmentDataHandler
Name = ContentTypeMap_1 Default Value = ContentType=application/xml; MimeType=text/xml;CharSet=UTF-8; ConvertAttachment=true;
Name = ContentTypeMap_2 Default Value = ContentType=text/xml; MimeType=text/xml;CharSet=UTF-8;
Name = PayloadDataHandlerMimeType Default Value = text/xml

Figure 6. Sample child meta-object for the Attachment data handler

Updating the top-level data-handler meta-object

A WebSphere Business Integration Adapter (such as the Adapter for JMS) uses the *MO_DataHandler_Default* meta-object to identify the data handlers it can use. WebSphere Server Access uses the *MO_Server_DataHandler* meta-object for the same purpose. Add a reference to the Attachment data handler in one of these meta-objects.

To associate MIME types that documents contain with the data handlers that provide support for these MIME types, an ICS-compatible component refers to a top-level data-handler meta-object. Table 42 summarizes the names of the top-level

meta-object, depending on the component that needs access to a data handler.

Table 42. Top-level data-handler meta-objects for InterChange Server

WebSphere Business Integration component	ICS-compatible component	Top-level data-handler meta-object
Adapter	Adapter for XML, Adapter for HTTP, Adapter for JMS	MO_DataHandler_Default
Access client using Server Access	Connect Servlet	MO_Server_DataHandler

To the MO_Server_DataHandler or MO_DataHandler_Default meta-object, you make the following modifications:

- Add an attribute whose name identifies the MIME type associated with the Attachment data handler instance; that is, for a document that contains this MIME type, the associated data handler can handle its conversion to a business object.

The attribute type of this attribute is the business object definition for the Attachment data handler's child meta-object (see "Creating the Attachment child meta-object" on page 56).

- Add an attribute for each of the supported attachment MIME types, if these do not already exist in the top-level data-handler meta-object.

The attribute type of these attributes would be the child meta-object of the associated data handler.

For example, suppose you have the Attachment data handler as configured in Figure 6 on page 58. Figure 7 shows the MO_DataHandler_Default meta-object with an attribute that associates the `wbic_attachment` MIME type with the instance of the Attachment data handler that is configured by the `MO_DataHandler_DefaultAttachmentConfig` child meta-object. This top-level data-handler meta-object also associates the document MIME type (`text/xml`) with the XML data handler's child meta-object.

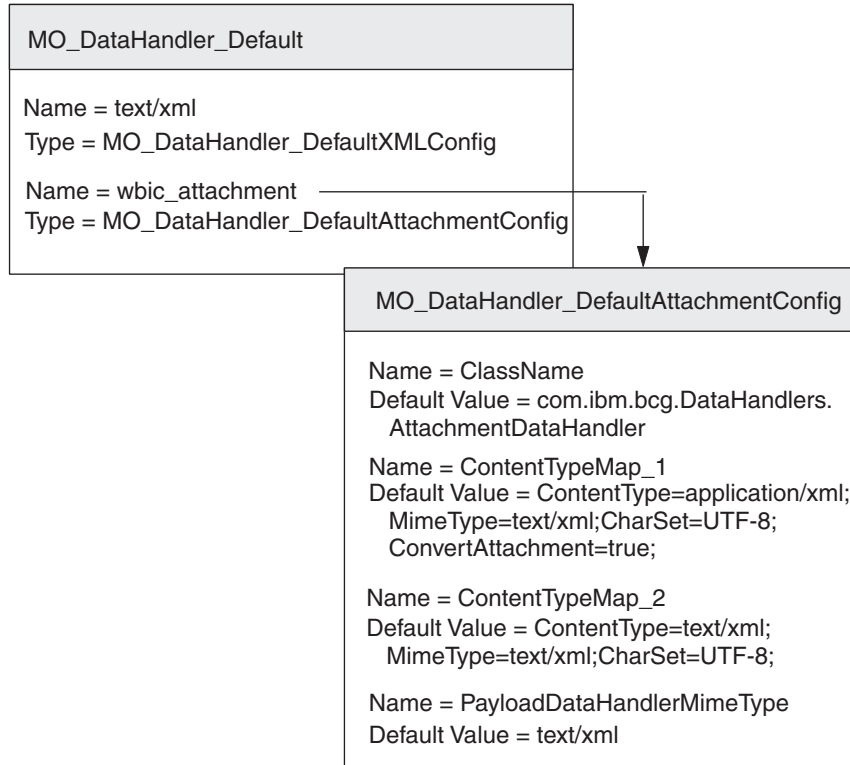


Figure 7. Associating the *wbic_attachment* MIME type with the Attachment data handler

For each unique combination of supported content types that you need to support, repeat the process by adding an attribute in the appropriate top-level data-handler meta-object, whose attribute name is the MIME type associated with the Attachment data handler instance and whose type is the name of the associated child meta-object. Also ensure that the configured MIME types (and their child meta-objects) exist in the top-level meta-object.

Creating attachment-related business object definitions

If you are sending or receiving documents that are wrapped in an XML transport envelope, your payload business object needs to contain attachment information. In any document flow, there is one payload and, optionally, multiple attachments. The Attachment data handler expects this attachment information to be in **attachment-related business objects**. Therefore, you must create business object definitions to represent this information. A business object definition is the form of information that InterChange Server uses. You use the Business Object Designer tool to create business object definitions.

Figure 8 shows the business-object structure for a payload that is wrapped in an XML transport envelope.

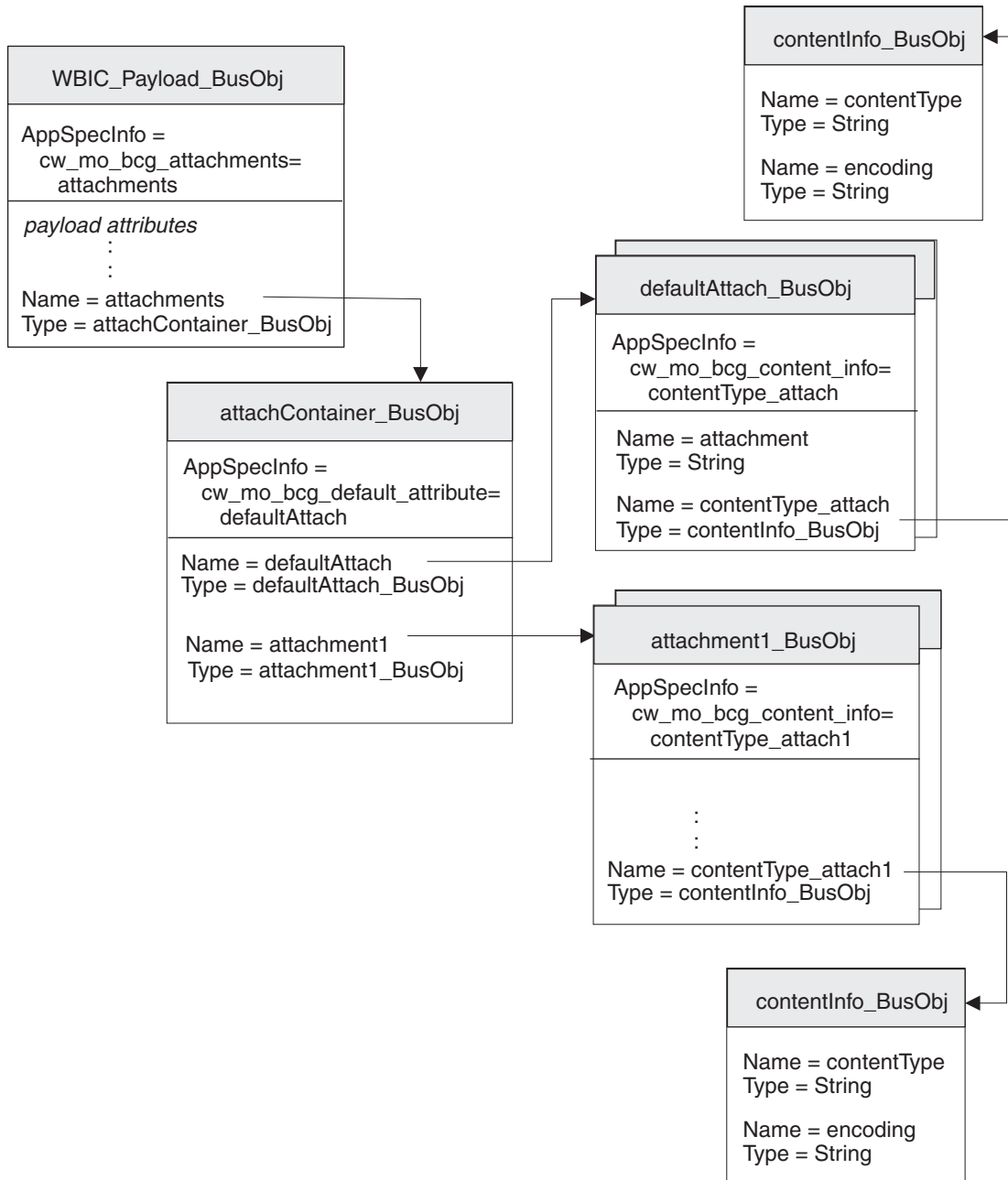


Figure 8. Relationship of the payload business object to the attachment business objects

As Figure 8 shows, all the attachments are contained in the attachment-container business object. If there are attachments, the payload business object has an attribute that corresponds to the attachment-container business object

Make sure your business-object structure includes attachment-required business objects by taking the following steps:

1. Create a business object definition to hold the content-type-encoding properties required by the Backend Integration packaging.
2. Create a business object definition for each type of attachment.
3. Create a business object definition for the attachment-container business object.

4. Modify the business object definition for your payload business object.

Each of these steps is described in the sections below.

Representing the content information

To store the content type and encoding of the associated payload or attachment, you create the **content-information business object**. To create a content-information business object definition, create the attributes shown in Table 43.

Table 43. Attributes of the content-information business object

Attribute	Attribute type	Description	Is key attribute?
contentType	String	The content type for the associated payload or attachment.	Yes
encoding	String	The character encoding for the associated payload or attachment	No

In Figure 8 on page 61, the `contentInfo_BusObj` business object definition contains attributes for the content type and encoding of the attachment. These attributes all have attribute-level application-specific information to specify the name of the related protocol header. For example, the `x-aux-sender-id` attribute has the application-specific information set as follows:

```
name=x-aux-sender-id
```

You can choose whatever name you want to identify the content-information business object definition. The application-specific information of the attachment business object determines if this is a Content Type Encoding business object type. Figure 8 on page 61 shows an example of a content-type-encoding business object definition called `contentType_BusObj`.

Representing attachment data

For attachment data that is not to be converted into a business object, you create the **default attachment business object**. This business object is useful for containing base64-encoded data that comes from the transport envelope.

To create a default-attachment business object definition, take the following steps:

- Create the attributes shown in Table 43.
- If you create a content-information business object, in the application-specific information for the default attachment business object definition, add the `cw_mo_bcg_content_info` tag to identify the attribute that contains the content information.

This `cw_mo_bcg_content_info` tag has the following format:

```
cw_mo_bcg_content_info=contentInfoAttr
```

where `contentInfoAttr` is the name of the attribute that contains the attachment-container business object.

Table 44. Attributes of the default attachment business object

Attribute	Attribute type	Description	Is key attribute?
attachment	String	The piece of attachment data. Note: This attribute is the key attribute of the business object definition.	Yes
An attribute to hold the content information	Business object	An optional attribute to hold the content -information business object, which provides the content type and encoding for the attachment data. This attribute should have single cardinality. Note: If this attribute does <i>not</i> exist, the Attachment data handler does not set the attachment data in the business object. For more information on the format of the content-information business object, see “Representing the content information” on page 62.	No

In Figure 8 on page 61, the defaultAttach_BusObj business object definition contains attributes for the piece of attachment data, including a content-information business object to hold its content type and encoding. The piece of attachment data that this default attachment business object represents does have a content-type encoding, represented by contentType_attach attribute. Therefore, the default attachment business object definition includes the following tag in its business-object-level application-specific information:

```
cw_mo_bcg_content_info=contentType_attach
```

Representing the attachments

For each kind of attachment in your document that converts to a business object, you must create a separate **attachment business object definition**. The attachment business object definition represents the actual data in a document attachment. To create an attachment business object definition, take the following steps:

1. Create an attribute for each piece of attachment data.
Possible attribute types can include String (for simple pieces of data) or a business object definition (for complex data).
2. If the attachment requires content-type encoding:
 - Create a content-type-encoding attribute.
The attribute type for this attribute is the content-type-encoding business object definition (see “Representing the content information” on page 62).
 - Add to the business-object-level application-specific information of the attachment business object definition the cw_mo_bcg_content_info tag, to identify the attribute that contains the content-type encoding.

This cw_mo_bcg_content_info tag has the following format:

```
cw_mo_bcg_content_info=contentTypeEncodingAttr
```

where *contentTypeEncodingAttr* is the name of the attribute that contains the content-type-encoding business object.

In Figure 8 on page 61, the payload document has one attachment, represented by the attachment1_BusObj business object definition. This attachment does have a content-type encoding, represented by contentTypeEncoding attribute. Therefore, the attachment business object definition includes the following tag in its business-object-level application-specific information:

```
cw_mo_bcg_content_info=contentTypeEncoding
```

Representing the attachment container

The attachment container contains all document attachments in the transport envelope. To represent the attachment container for InterChange Server, you create the **attachment-container business object**. Each attribute in the attachment-container business object represents one attachment.

To create the attachment-container business object definition, take the following steps:

1. Add an attribute for each attachment in the document that is to be converted to a business object.

The attribute type for each of these attributes is the associated attachment business object (see “Representing the attachments” on page 63). Each attribute should have multiple cardinality.

2. Add to the application-specific information for each attribute the `wbic_type` tag to identify the attribute as an attachment.

The `wbic_type` tag has the following format:

```
wbic_type=Attachment
```

Note: An attachment attribute can have multiple cardinality.

3. If the payload contains attachment data that should *not* be converted to a business object:

- Add an attribute for the default attachment business object.

The attribute type for this attribute is the default attachment business object (see “Representing attachment data” on page 62). It is the key attribute for the attachment-container business object. This attribute does *not* require the `wbic_type` tag in its application-specific information.

Note: The attachment-container business object can contain only *one* default attachment attribute. However, this attribute can have multiple cardinality.

- Add to the business-object-level application-specific information of the attachment business object definition the `cw_mo_bcg_default_attribute` tag, to identify the attribute that contains the attachment data.

This `cw_mo_bcg_default_attribute` tag has the following format:

```
cw_mo_bcg_content_info=defaultAttachmentAttr
```

where `defaultAttachmentAttr` is the name of the attribute that contains the default attachment business object.

Important: If no default-attachment attribute exists, the Attachment data handler can *not* convert any attachments that do not have an associated content-type map or attachments that are not converted to business objects. These attachments will be lost during the conversion to business-object representation.

In Figure 8 on page 61, the attachment container is represented by the `attachContainer_BusObj` business object definition. This attachment-container business object definition has the following attributes:

- The `attachment1` attribute represents the single attachment for the document. Therefore, the attachment-container business object definition includes the following tag in its attribute-level application-specific information:

```
wbic_type=Attachment
```

This attachment is represented by the `attachment1_BusObj` business object definition.

- The `defaultAttach` attribute represents the attachment data that does *not* require conversion to the business-object representation. Therefore, the attachment-container business object definition includes the following tag in its business-object-level application-specific information:

```
cw_mo_bcg_default_attribute=defaultAttach
```

Modifying the payload business object definition

The payload business object definition represents the information in your document. It contains an attribute for each piece of information you are transferring between Business Integration Connect and InterChange Server. For information on the creation of the payload business object definition, see “Business object for the document” on page 45.

If you are sending or receiving documents that contain attachments, your payload business object needs to contain attachment information. In any document flow, there is one payload and, optionally, multiple attachments. If the payload of your document contains attachments, you must modify the payload business object definition as follows:

- Create an attribute to hold the payload data.
You might find it easier to use if your actual payload data is stored in a separate payload business object definition. In this case, the top-level payload business object contains an attribute for the payload data whose attribute type is the business object definition of actual payload data.
- Add an attachment container:
 - Add an attribute to hold the attachment container.
The attribute type of this attribute is the attachment-container business object definition (see “Representing the attachment container” on page 64). This attribute should have single cardinality.
 - In the application-specific information for the payload business object definition, add the `cw_mo_bcg_attachment` tag to identify the attribute that contains the attachment container.
This `cw_mo_bcg_attachment` tag has the following format:

```
cw_mo_bcg_attachment=attachContainerAttr
```

where `attachContainerAttr` is the name of the attribute that contains the attachment-container business object.
- Optionally, you can specify the content type of the payload. The Attachment data handler uses this content type to determine which data handler to instantiate to convert the payload data. If it finds a matching content type in the content-type maps, it instantiates the data handler for this content type.
 - Add a content-information attribute, which is an optional attribute to hold the content type and encoding for the payload. This attribute should have single cardinality.

Note: If this attribute does *not* exist, the Attachment data handler obtains the data handler to convert the payload from the setting of the `PayloadDataHandlerMimeType` configuration property, in its child meta object.
 - In the application-specific information for the payload business object definition, add the `cw_mo_bcg_content_info` tag to identify the attribute that contains the content information.

This `cw_mo_bcg_content_info` tag has the following format:

```
cw_mo_bcg_attachment=contentInfoAttr
```

where *contentInfoAttr* is the name of the attribute that contains the content-information business object. For more information on the format of the content-information business object, see “Representing the content information” on page 62.

- Any configuration attributes required for your transport protocol.

For example, if you are using the JMS transport protocol, your payload business object definition must contain the JMS dynamic business object. For more information, see the section on how to create business object definitions in support of your transport protocol.

Chapter 3. Integrating InterChange Server over HTTP

This chapter describes how to integrate WebSphere Business Integration Connect with WebSphere InterChange Server over the HTTP transport protocol. It provides information on how to configure InterChange Server (ICS) and the ICS-compatible components required for communication over HTTP.

Note: For information on how to configure WebSphere Business Integration Connect to communicate with InterChange Server over HTTP, see “Configuring Business Integration Connect for InterChange Server” on page 41. For general information on how to configure InterChange Server, see “Configuring InterChange Server” on page 44.

This chapter provides the following information on how to send and receive documents between WebSphere Business Integration Connect and WebSphere InterChange Server through the use of the HTTP transport protocol:

- “Using HTTP transport protocol with pre-4.2.2 ICS”
- “Using HTTP transport protocol with v4.2.2 ICS” on page 95
- “Sending SOAP documents over HTTP/S” on page 110

Using HTTP transport protocol with pre-4.2.2 ICS

WebSphere Business Integration Connect can send and receive documents with a pre-4.2.2 version of WebSphere InterChange Server (ICS) over the HTTP transport protocol.

Notes:

1. To send and receive documents between WebSphere Business Integration Connect and a *version 4.2.2* InterChange Server over the HTTP transport protocol, see “Using HTTP transport protocol with v4.2.2 ICS” on page 95.
2. If you are exchanging SOAP documents over the HTTP transport protocol, see “Sending SOAP documents over HTTP/S” on page 110.

This section provides the following information on how to configure a pre-4.2.2 InterChange Server and the appropriate ICS-compatible components for use with Business Integration Connect over HTTP:

- “Sending documents to pre-4.2.2 ICS through HTTP”
- “Receiving documents from pre-4.2.2 ICS through HTTP” on page 81
- “Creating business object definitions for pre-4.2.2 ICS over HTTP” on page 87
- “Creating pre-4.2.2 ICS artifacts for HTTP” on page 95

Sending documents to pre-4.2.2 ICS through HTTP

This section provides the following information to describe how to send documents from Business Integration Connect to a pre-4.2.2 ICS over the HTTP transport protocol:

- “Components required for sending” on page 68
- “Configuring the Connect Servlet” on page 71
- “Configuring the Wrapper data handler” on page 76
- “Creating business object definitions for sending documents” on page 80

The document that Business Integration Connect sends to InterChange Server initiates event notification within InterChange Server.

Components required for sending

Business Integration Connect can send documents to the following pre-4.2.2 versions of InterChange Server over the HTTP transport protocol:

- Version 4.1.1
- Version 4.2.0
- Version 4.2.1

For Business Integration Connect to send a document to a pre-4.2.2 ICS using the HTTP transport protocol requires that these two components be configured. Table 45 summarizes these configuration steps.

Table 45. Configuring Business Integration Connect and InterChange Server

Component	Version	For more information
WebSphere Business Integration Connect	4.2.2	“Configuring for outgoing documents over HTTP transport protocol” on page 41 “Configuring for incoming documents over HTTP transport protocol” on page 43
WebSphere InterChange Server	4.1.1, 4.2.0, 4.2.1	“Creating pre-4.2.2 ICS artifacts for HTTP” on page 95

In addition, to send a document to ICS over the HTTP transport, you use the ICS-compatible components listed in Table 46.. Most of these components are provided as part of the Business Integration Connect release.

Table 46. Components required to send documents to pre-4.2.2 ICS through HTTP

Component	Description	Notes and restrictions
WebSphere Business Integration Connect Servlet (Connect Servlet)	This servlet is a WebSphere InterChange Server access client. An access client is a process that is external to InterChange Server (ICS) and can request execution of a collaboration within ICS.	The servlet can be used with the pre-4.2.2 versions of WebSphere InterChange Server. Note: The servlet <i>cannot</i> be used with version 4.2.2 WebSphere InterChange Server.
Wrapper data handler	This data handler is called by the Connect Servlet to convert the HTTP message into the appropriate data business object. It invokes the data handler that is appropriate for your message. For example, if the payload is formatted in XML, the Wrapper data handler can be configured to call the Data Handler for XML.	<i>None</i>
A payload data handler	This data handler converts the document payload between its document format (usually XML) and its business-object representation.	This data handler is required and must support the MIME type of your payload document.
Attachment data handler	This data handler handles attachment documents for your document message.	This data handler is required <i>only</i> if your documents include attachments.

Note: All components listed in Table 46 are included on the Business Integration Connect installation medium. For information on the location of these components, see “Deploying the Connect Servlet” on page 72.

Figure 9 provides an overview of how Business Integration Connect sends documents to a pre-4.2.2 ICS over the HTTP transport protocol.

Note: The Wrapper data handler, the Attachment data handler, and the payload data handler all execute within InterChange Server.

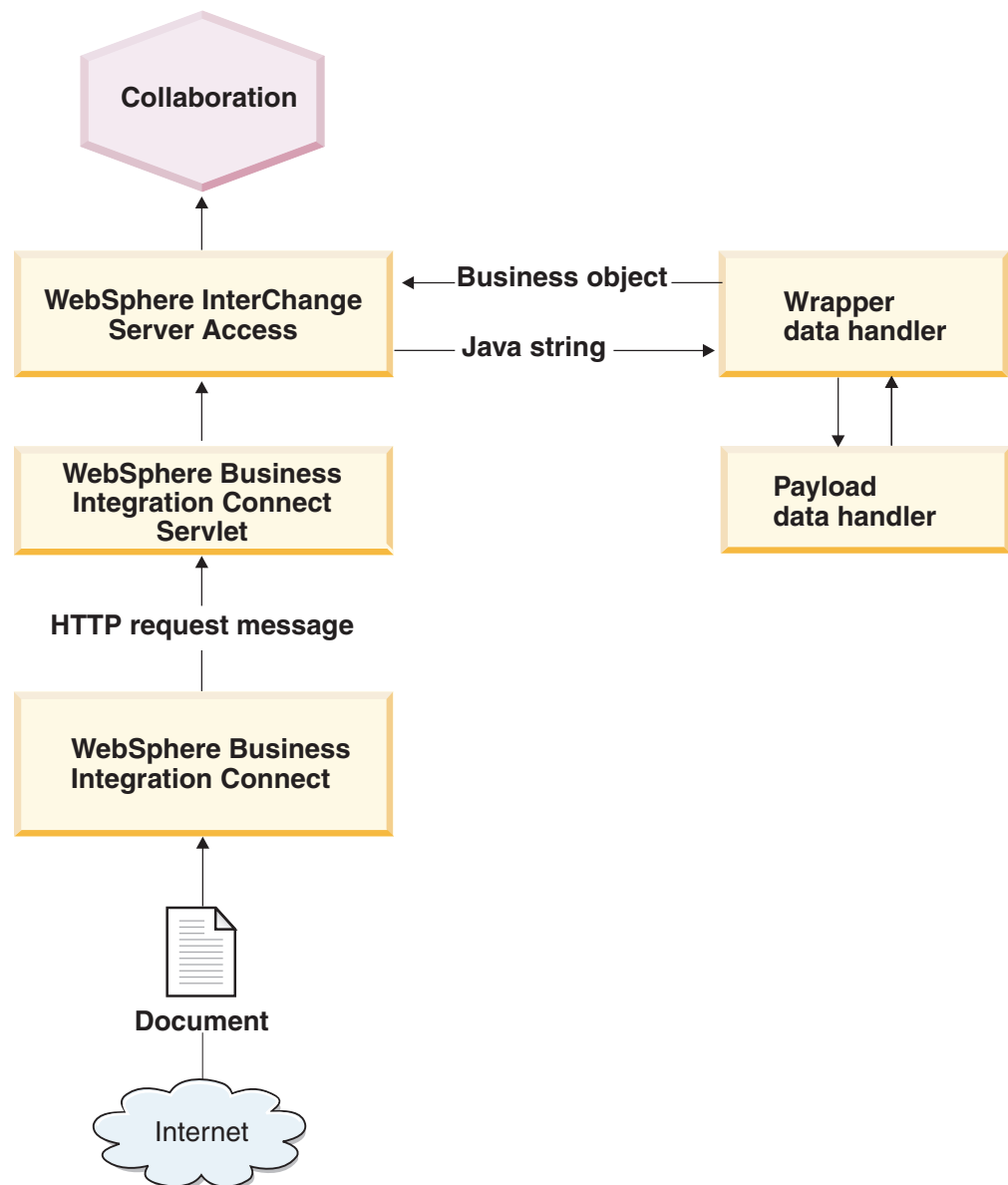


Figure 9. Message flow from Business Integration Connect to a collaboration through the HTTP transport protocol

As Figure 9 shows, the WebSphere Business Integration Connect Servlet is the ICS-compatible component with which Business Integration Connect interacts directly. This Connect Servlet is an access client, which is a process external to InterChange Server that can request execution of an ICS collaboration. The access client issues calls from an application programming interface (API) called the

Server Access Interface to interact with ICS. These calls are received and interpreted by WebSphere InterChange Server Access, which is the component within ICS that handles interactions with access clients. The Server Access Interface invokes the collaborations synchronously.

Notes:

1. Even though some interactions between Business Integration Connect and back-end systems are asynchronous, Server Access still invokes the collaboration synchronously and waits until the collaboration execution has completed.
2. For more detailed information on access clients and Server Access, see the *Access Development Guide* in the WebSphere InterChange Server documentation set.

The following steps describe how Business Integration Connect participates in event notification by sending a document to a collaboration within ICS over the HTTP transport protocol:

1. Business Integration Connect invokes the WebSphere Business Integration Connect Servlet to send the document to InterChange Server.
Business Integration Connect sends the document to the URL specified as the target gateway.

Note: The Connect Servlet can be used to invoke multiple collaborations.

2. The Connect Servlet creates a Java string from the HTTP request message that Business Integration Connect sends.

The HTTP request message contains two parts:

- HTTP transport-protocol headers:
 - The standard headers
 - The custom headers, which Business Integration Connect sets if Backend Integration packaging was specified for the document
- The message, whose format depends on the type of packaging that is used

3. The Connect Servlet checks its servlet properties file to determine the collaboration to invoke, along with the verb and MIME type to use.

Each URL corresponds to a collaboration to be invoked. (See “Configuring the Connect Servlet” on page 71 for information.)

4. The Connect Servlet sends the Java string, along with the information from the servlet properties file, to WebSphere InterChange Server Access using calls in the Server Access Interface.

Because the Connect Servlet can only send a document to InterChange Server (it cannot receive a document), it can only participant in event notification with InterChange Server.

Note: To support request processing with InterChange Server, Business Integration Connect must interact with the WebSphere Business Integration Adapter for XML. For more information, see “Receiving documents from pre-4.2.2 ICS through HTTP” on page 81.

5. WebSphere InterChange Server Access, within InterChange Server, receives the Java string and invokes the Wrapper data handler.

The job of the Wrapper data handler is to convert the Java string into the corresponding business-object structure. It is business objects that InterChange Server expects as input.

6. The Wrapper data handler takes the following step to convert the Java string to its business-object structure:
 - a. Extract the headers and payload from the Java string.

Note: If the document that Business Integration Connect sent includes attachments, the Wrapper data handler can be configured to call the Attachment data handler. The actions of the Attachment data handler are described in “Handling documents with attachments” on page 49.

- b. Check the MIME type of the payload and call the data handler that has been configured for that MIME type to convert the payload into a payload business object.
 - c. Create the HTTP-properties business object and the dynamic business object. Set the HTTP headers in the HTTP-properties business object, which is a child of the dynamic meta-object of this payload business object.
 - d. Create the top-level business object and set the event business object as its request business object.
7. Server Access invokes the collaboration, passing it the top-level business object. Make sure the collaboration port for the collaboration object you will be invoking is configured as the external port. Refer to the WebSphere InterChange Server documentation for details on how to configure ports.
8. The collaboration executes and returns the top-level business object to the Wrapper data handler.

The Wrapper data handler expects the payload business object to have a hierarchical structure. For information on the structure of this payload business object, see “Creating business object definitions for sending documents” on page 80.

- e. Return the top-level business object to the Server Access within InterChange Server.

Whether the response business object (within the top-level business object) is populated depends on the type of interaction between InterChange Server and Business Integration Connect, as follows:

- For *asynchronous* interactions, the collaboration should *not* populate the response business object.
- For *synchronous* interactions in which a response should be returned in the same HTTP connection, the collaboration should populate the response business object.

For more information, see “Response business object” on page 91.

9. If the interaction is successful, the Connect Servlet returns an HTTP 200 OK acknowledgment to Business Integration Connect.

Configuring the Connect Servlet

The WebSphere Business Integration Connect Servlet is an access client, which is a process that is external to InterChange Server and can request execution of a collaboration within InterChange Server. The access client uses calls from an application programming interface (API) called the Server Access Interface to interact with ICS. These calls are received and interpreted by WebSphere InterChange Server Access, which is the component of InterChange Server that handles interactions with access clients.

Note: For more detailed information on access clients and Server Access, see the *Access Development Guide* in the WebSphere InterChange Server documentation set.

Configuration of the Connect Servlet involves the following steps:

- “Deploying the Connect Servlet”
- “Setting the Connect Servlet properties” on page 73

Deploying the Connect Servlet: The Connect Servlet, Wrapper data handler, and the repository file for the Wrapper data handler are available on the Business Integration Connect installation medium, in the locations listed in Table 47.

Table 47. Location of the Connect Servlet components

Component	Location
Connect Servlet	Integration/WBI/WICS/WBICServlet/ bcgwbiservlet.war
Wrapper data handler	Integration/WBI/WICS/WBICServlet/ bcgwbirapperdh.jar
Repository file for Wrapper data handler	Integration/WBI/WICS/WBICServlet MO_DataHandler_WBIWrapper.in

Note: If you are expecting to send documents that include attachments, you can also deploy the Attachment data handler and its associated repository file, as described in “Deploying the Attachment data handler” on page 55.

This servlet can connect to WebSphere InterChange Server versions 4.1.1, 4.2.0, and 4.2.1. It can be deployed on the platforms on which any of these versions of InterChange Server is supported. Additionally you need to make sure that the Server Access Interface is supported on that platform. Refer to the WebSphere InterChange Server documentation for a list of the platforms on which the ICS version you are using is supported.

To deploy the components in Table 47, follow these steps:

1. Deploy the Connect Servlet and associated files into the Web server according to the documentation for the Web server.
2. Make sure the following files are in the CLASSPATH of the Connect Servlet:

- crossworlds.jar
- vbjorb.jar

These files can be found in the lib subdirectory of the InterChange Server product directory.

Notes:

- a. These files must be from the same version of the InterChange Server that you will be invoking.
 - b. These files must be available to the web container of the Connect Servlet in your web server. For more information on how to make files available to a web container, consult your web server documentation.
3. Make sure the following files are in the CLASSPATH of the Connect Servlet:

- mail.jar
- log4j-1.2.8.jar

These files can be found on the Business Integration Connect installation medium in the following directory:

integration/wbi/wics/http/lib/thirdparty

Note: These files must be available to the web container of the Connect Servlet in your web server. For more information on how to make files available to a web container, consult your web server documentation.

4. Make the InterChange Server Interoperable Object Reference (.ior) file available on the machine on which the Connect Servlet is deployed.

If the Connect Servlet is on a different machine from InterChange Server, you can take either of the following actions to make the .ior file available:

- Copy the file from the InterChange Server machine to the machine on which Business Integration Connect is installed.
- Put the file into a shared location, one that both InterChange Server and Business Integration Connect can access.

Note: You will also need to update the ICS_IORFILE property in the Connect Servlet's properties file with the location of this .ior file. For more information, see "Identifying the InterChange Server instance."

Setting the Connect Servlet properties: As mentioned in "Sending documents to pre-4.2.2 ICS through HTTP" on page 67, the **servlet properties file** contains information, such as port name and verb, that the WebSphere Business Integration Connect Servlet needs to invoke a collaboration. You must create this servlet properties file, specifying general information about WebSphere InterChange Server. Then, for any collaboration you want the servlet to invoke, you provide information about that collaboration.

This section provides the following information about how to set the properties for the Connect Servlet:

- "Creating the servlet property file"
- "Identifying the location of the servlet properties file" on page 76

Creating the servlet property file: A servlet properties file contains the sections listed in Table 48.

Table 48. Sections of the servlet properties file

Section of servlet properties file	Description	For more information
General information	Properties to identify the InterChange Server instance	"Identifying the InterChange Server instance"
Collaboration information	Properties to identify each collaboration to invoke	"Identifying the collaborations to invoke" on page 74
Logging information	Properties to configure the servlet's logging	"Specifying the location of the servlet log file" on page 75

Identifying the InterChange Server instance: The first section of the Connect Servlet's properties file contains general information to identify the InterChange Server instance with which Business Integration Connect communicates. This ICS instance contains the collaboration (or collaborations) that Business Integration Connect needs to invoke. Table 49 shows the general properties of the servlet properties file.

Table 49. General properties of the servlet properties file

Property name	Description	Example
ICS_SERVERNAME	The host machine on which WebSphere InterChange Server is running.	Server1
ICS_VERSION	The version number of WebSphere InterChange Server. Possible values are 4.1.1, 4.2.0, and 4.2.1.	4.2.0

Table 49. General properties of the servlet properties file (continued)

Property name	Description	Example
ICS_IORFILE	The file name of the Interoperable Object Reference (.ior) file, which is used to access WebSphere InterChange Server Access. The example shows how you would specify the path on a Windows system. Note: This path should be entered on one line.	c:/myiorlocation/Server1ICS.ior
ICS_USERNAME	The User ID for connecting to WebSphere InterChange Server.	admin
ICS_PASSWORD	The password for connecting to WebSphere InterChange Server.	null
ICS_ENCRYPTED_PASSWORD	An indication of whether the ICS_PASSWORD is encrypted. The servlet sets this field to true if the password is encrypted.	false
ICS_DISABLEENCRYPTION	An indication of whether password encryption is disabled (true) or enabled (false). Set this field to false if you want to allow passwords to be encrypted.	true

Note: For a sample servlet properties files that defines the values listed in the Example column of Table 49, see “Sample servlet properties file” on page 75.

Identifying the collaborations to invoke: The second section of the Connect Servlet’s properties file contains collaboration information, which associates the collaboration URL with associated collaboration properties. This section identifies collaboration URLs in two parts, as follows:

- The `WBIC_SERVLET_COUNT` property specifies the number of collaborations that are identified in this collaboration section of the servlet properties file. It specifies an integer number of URLs configured in this file:
 - If it is set to 1, the Connect servlet will process the URL defined with the `WBIC_URL_1` property.
 - If it is set for 2, the Connect servlet will process the URLs defined with *both* the `WBIC_URL_1` and `WBIC_URL_2` properties.
- The `WBIC_URL_count` property identifies the relative URL for the collaboration. The associated collaboration properties have property names of the form `WBIC_URL_count_propertyName`. Table 50 defines these a sample of these `WBIC_URL_count_propertyName` properties. In the Example column, this table provides sample values of the `WBIC_URL_count_propertyName` properties for the first collaboration URL (*count* is 1).

Table 50. Collaboration properties of the servlet properties file

Property name	Description	Example
WBIC_SERVLET_COUNT	The number of URLs configured in this file: <ul style="list-style-type: none"> • If it is set to 1, the servlet will process the URL and properties for <code>WBIC_URL_1</code>. • If it is set for 2, the servlet will process the URL and properties for <code>WBIC_URL_1</code> and <code>WBIC_URL_2</code>. 	1
WBIC_URL_1	The name of the relative URL	PurchaseOrder
WBIC_URL_1_COLLAB	The name of the collaboration	PurchaseOrderCollab
WBIC_URL_1_PORT	The port name of the collaboration	From
WBIC_URL_1_VERB	The verb subscribed to by the collaboration	Create
WBIC_URL_1_WRAPPER_MIME	The MIME type that the Wrapper data handler supports. Note that the example is in lowercase.	wbic/wrapper
WBIC_URL_1_CHARENCODE	The character encoding to use for the HTTP requests. Specify valid Java character encoding.	UTF-8

Note: For a sample servlet properties files that defines the values listed in the Example column of Table 50, see “Sample servlet properties file.”

The collaboration section of the servlet properties file provides a relative URL to identify the collaboration to execute. To look up the collaboration at run-time, the Connect Servlet combines the following pieces of information:

- The URL that identifies the location of the Connect Servlet
- The relative URL, specified for the collaboration in the Connect Servlet properties file

For example, if you used the sample values shown in Table 50 on page 74, the Connect Servlet would need to obtain the URL of the PurchaseOrderCollab collaboration. To look up this URL, the servlet takes the following steps:

1. Obtain the servlet URL, which identifies the location of the Connect Servlet.

The servlet obtains the servlet URL from your web server. For example, suppose you deployed the Connect Servlet at the following location:

```
http://www.yourcompany.com/tasks
```

2. Append to the servlet URL the path in the `WBIC_URL_count` property.

In Table 50 on page 74, the `WBIC_URL_1` property contains the value “PurchaseOrder”. Therefore, the Connect Servlet would append this string to the servlet URL to obtain the following URL for the collaboration:

```
http://www.yourcompany.com/tasks/PurchaseOrder
```

In the collaboration properties, the `WBIC_URL_1_WRAPPER_MIME` property specifies the MIME type for the Wrapper data handler. If you specify more than one MIME type, you need multiple meta-objects. See “Creating the Wrapper child meta-object” on page 77 for information.

Specifying the location of the servlet log file: You specify logging properties in the third section of the Connect Servlet’s properties file. You specify the location of the servlet log file in the properties file by adding the following statement:

```
log4jappender.RollingFile.File=logFileLocation
```

As shown in Figure 10 on page 76, the `log4jappender.RollingFile.File` property is in the section of the servlet properties file that configures the Log4J. To configure the Connect Servlet, you only need to specify the location of the log file, by setting the `log4jappender.RollingFile.File` property. If you are familiar with Log4J, you can set other of its properties as well.

Sample servlet properties file: Figure 10 on page 76 shows an example of the servlet properties file, which configures the values from the Example columns of Table 49 and Table 50.

```

# Example properties file for WebSphere Business Integration
# Connect Servlet
ICS_SERVERNAME=Server1
ICS_VERSION=4.2
ICS_IORFILE=C:/myiorlocation/Server1InterChangeServer.ior
ICS_USERNAME=admin
ICS_PASSWORD=null
ICS_ENCRYPTED_PASSWORD=false
ICS_DISABLEENCRYPTION=true

# Collaboration properties for single collaboration
WBIC_SERVLET_COUNT=1

WBIC_URL_1=PurchaseOrder
WBIC_URL_1_COLLAB=PurchaseOrderCollab
WBIC_URL_1_CHARENCODE=UTF-8
WBIC_URL_1_PORT=From
WBIC_URL_1_VERB=Create
WBIC_URL_1_WRAPPER_MIME=wbic/wrapper

#Log4J Debug Properties
#Possible Categories - debug/info/warn/error/fatal
#Default Category "error". Output to: stdout and RollingFile
log4j.rootCategory=debug,RollingFile
log4j.appender.RollingFile=org.apache.log4j.RollingFileAppender

#Log File Name
log4j.appender.RollingFile.File=D:\\_DEV\\servlet.log
log4j.appender.RollingFile.MaxFileSize=1000KB

#Number of backup files to keep
log4j.appender.RollingFile.MaxBackupIndex=10
log4j.appender.RollingFile.layout=org.apache.log4j.PatternLayout
log4j.appender.RollingFile.layout.ConversionPattern=
    %d{yyyy-MM-ddHH:mm:ss} %-5p [%c{1}] - %m%n

```

Figure 10. Sample servlet properties file

You can also find a sample servlet properties file in the SAMPLES directory on the Business Integration Connect installation medium.

Identifying the location of the servlet properties file: The deployment descriptor of Connect Servlet, `web.xml`, provides initialization parameters for the servlet. To identify the location of the servlet properties file, you set the `WBIC_FILENAME` parameter in this deployment descriptor. This parameter specifies the absolute pathname of the Connect servlet properties file.

For example, if the sample servlet properties file shown in Figure 10 was called `connectServlet.cfg` and it was located in the deployment directory of the Connect Servlet (for example, `C:\WBIC\integration`), you would need to set the `WBIC_FILENAME` parameter as follows:

```
C:\WBIC\integration\connectServlet.cfg
```

Configuring the Wrapper data handler

The Wrapper data handler converts a document from its serialized format (which the Connect Servlet has created from the HTTP message) into its corresponding business object. When the Connect Servlet invokes a collaboration, it sends to InterChange Server the serialized format of the document that Business Integration Connect has sent it. This collaboration request is received by WebSphere Server Access, which resides within InterChange Server. As Figure 9 on page 69 shows, Server Access calls the Wrapper data handler, passing it the Business Integration Connect document. The data handler returns the corresponding payload business object.

To configure the Wrapper data handler, take the following steps:

- “Specifying the location of the Wrapper data handler”
- “Creating the configuration business objects for the Wrapper data handler”

The steps to configure the Wrapper data handler are summarized in the following sections. For general information about data handlers, see the *Data Handler Guide* in the WebSphere InterChange Server documentation set.

Specifying the location of the Wrapper data handler: InterChange Server needs to know the location of the Wrapper data handler, so that it can load it at run-time. To specify the location, take the following steps:

1. Edit the ICS startup script, `start_server.bat`, which is located in the `bin` subdirectory of the InterChange Server product directory (on the machine where InterChange Server resides).
2. To this file, add the jar file for the Wrapper data handler, `bcgwbwrapperdh.jar`, to the list of jar files included at ICS startup. Usually, data-handler jar files are added to the `DATAHANDLER` variable in the ICS startup script.

Note: If you have installed the optional Attachment data handler, you must also add its jar file to the ICS startup script. For more information, see “Specifying the location of the Attachment data handler” on page 55.

Creating the configuration business objects for the Wrapper data handler: To identify the data handler to invoke, Server Access (within InterChange Server) checks the top-level data-handler meta-object, `MO_Server_DataHandler`. This file is located in the following subdirectory of the InterChange Server product directory:
`repository\edk`

This top-level meta-object associates a MIME type with a child meta-object, which contains the configuration information for the data handler. Therefore, creating configuration business involves the following steps:

1. “Creating the Wrapper child meta-object”
You must initialize a child meta-object with the Wrapper data handler’s configuration information.
2. “Editing the `MO_Server_DataHandler` meta-object” on page 79
You must create an entry in this meta-object that associates a MIME type with the name of the Wrapper data handler’s child meta-object.

Creating the Wrapper child meta-object: To configure the Wrapper data handler, you must create its child meta-object and initialize it with configuration information. The data handler uses the attributes of this meta-object to obtain its configuration information, including the name of the data-handler class to instantiate. To create this meta-object, you create a business object definition that contains the attributes listed in Table 51.

Note: Use Business Object Designer to create this business object definition.

Table 51. Configuration properties in the Wrapper child meta-object

Attribute	Description
ClassName	Class name (required), which points to the following data handler class: <code>com.ibm.bcg.integration.wbi.datahandlers.WBICWrapperDataHandler</code>
TopBOPrefix	Prefix is used to determine the name of the top-level business object. If the request business object returned by the data handler configured for the request does <i>not</i> have the <code>wbic_mainoname</code> tag in its business-object-level application-specific information, the name of the top-level object is obtained by adding the <code>TopBOPrefix</code> to the name of the request business object.
wbic_request_mime	MIME type supported by the data handler that the Wrapper data handler invokes to process the payload of the request message. Make sure that this data handler has been configured so that it can be invoked by the WebSphere InterChange Server Access. For more information, see “Editing the <code>MO_Server_DataHandler</code> meta-object” on page 79. Note: If your documents contain attachments, the MIME type for this configuration property should be the MIME type that invokes the Attachment data handler. For more information, see “Handling documents with attachments” on page 49.
wbic_response_mime	MIME type of the data handler that the Wrapper data handler will invoke to process the payload of the response message. Note: You do not have to set the <code>wbic_response_mime</code> if Business Integration Connect is not expecting a response.

Important: To assign a value to the attributes in Table 51., set the default value of the attribute. For example, if the Wrapper data handler is to use the Delimited data handler for its request message, set the Default Value of the `wbic_request_mime` attribute to `text/delimited`.

You can define a child meta-object for *each* instance of the Wrapper data handler that you need to use. For example, if you need to support only one request MIME type or one combination of request and response MIME types, you can create a single child meta-object and set the default values of the `wbic_request_mime` and `wbic_response_mime` attributes accordingly. If, however, you need to support different combinations of request and response MIME types, you can create one child meta-object for each of the supported combinations.

Business Integration Connect provides the following InterChange Server repository file, which contains a sample child meta-object for the Wrapper data handler:

```
ProductDir/Integration/WBI/WICS/WBICServlet/MO_DataHandler_WBICWrapper.in
```

where *ProductDir* is the directory of your installed Business Integration Connect product. This repository file defines a single instance of the Wrapper data handler, which is configured to call the Delimited data handler for both request and response business objects. Figure 11 shows the sample child meta-object called `MO_DataHandler_WBICWrapper`.

MO_DataHandler_WBICWrapper
Name = ClassName Default Value = com.ibm.bcg.integration.wbi. datahandlers.WBICWrapperDataHandler
Name = TopBOPrefix Default Value = WBIC
Name = wbic_request_mime Default Value = text/delimited
Name = wbic_response_mime Default Value = text/delimited

Figure 11. Sample child meta-object for a Wrapper data handler

If you also needed to support a document whose request message was in XML, you would create a second child meta-object to represent a second instance of the Wrapper data handler. In this child meta-object, the default value of the `wbic_request_mime` attribute would have the MIME type of `text/xml`.

Editing the MO_Server_DataHandler meta-object: WebSphere InterChange Server Access uses a top-level meta-object, called `MO_Server_DataHandler`, to associate MIME types that access clients can handle with the data handlers that provide the support for these MIME types. Specifically, this top-level meta-object associates MIME types with data-handler child meta-objects.

The `MO_Server_DataHandler` meta-object is a business object definition. Therefore, to edit this meta-object, bring up the `MO_Server_DataHandler` in Business Object Designer and add to it a new attribute for each supported instance of the Wrapper data handler. Each instance of this data handler is a unique combination of request and response MIME types.

To the `MO_Server_DataHandler` meta-object, you make the following modifications:

- Add an attribute whose name identifies the MIME type associated with the Wrapper data handler instance; that is, for a document that contains this MIME type, the associated data handler can handle its conversion to a business object. The attribute type of this attribute is the business object definition for the Wrapper data handler's child meta-object (see "Creating the Wrapper child meta-object" on page 77).
- Add an attribute for each of the supported request and response MIME types, if these do not already exist in the `MO_Server_DataHandler` meta-object. The attribute type of these attributes would be the child meta-object of the associated data handler.

For example, suppose you have the Wrapper data handler as configured in Figure 11.. Figure 12 shows the `MO_Server_DataHandler` meta-object with an attribute that associates the `wbic_wrapper` MIME type with the instance of the Wrapper data handler that is configured by the `MO_DataHandler_WBICWrapper` child meta-object. This `MO_Server_DataHandler` meta-object also associates the request and response MIME types (`text/delimited`) with the Delimited data handler's child meta-object.

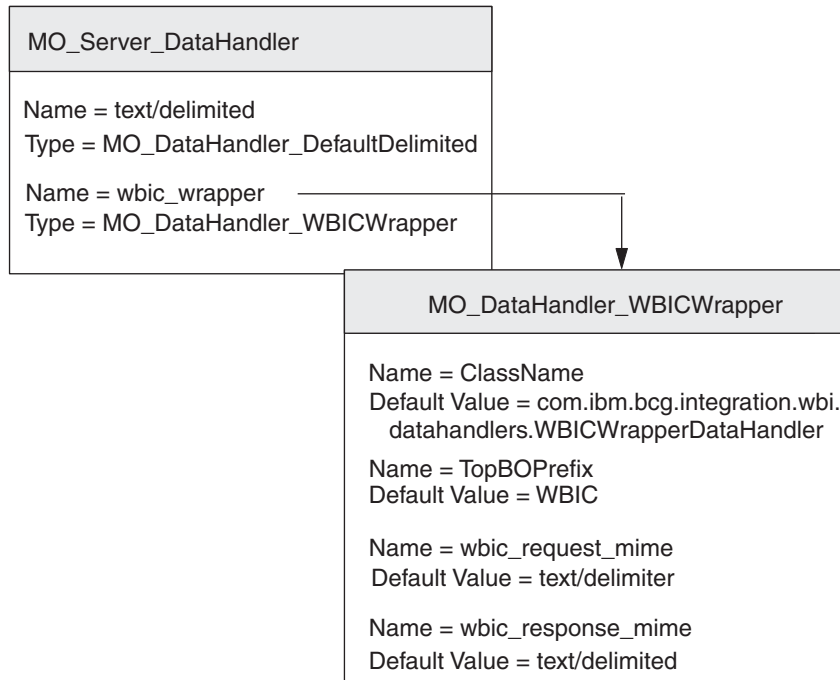


Figure 12. Associating the *wbic_wrapper* MIME type with the Wrapper data handler

For each unique combination of request and response MIME type that you need to support, repeat this process by adding an attribute in the `MO_Server_DataHandler` top-level meta-object whose attribute name is the MIME type associated with the Wrapper data handler instance and whose type is the name of the associated child meta-object. Also ensure that the configured request and response MIME types (and their child meta-objects) exist in `MO_Server_DataHandler`.

Note: If you are using the Attachment data handler to process attachments in your Business Integration Connect documents, you must also modify the `MO_Server_DataHandler` to support the Attachment data handler, as described in “Configuring the Attachment data handler” on page 56.

Creating business object definitions for sending documents

The WebSphere Business Integration Connect Servlet sends your document to InterChange Server in the form of a payload business object. For the Connect Servlet, the payload business object is represented as a hierarchy of business objects. The Wrapper data handler creates this business-object hierarchy when it receives a Business Integration Connect document. Therefore, you must create business object definitions to represent this hierarchy.

Because the Connect Servlet participants *only* in event notification with InterChange Server, the request and response attributes of the top-level business object are interpreted as shown in Table 52.

Table 52. Request and response business objects in event notification

Attribute	Use
Request business object	Contains the request message from Business Integration Connect; this message is the event that triggers the collaboration.

Table 52. Request and response business objects in event notification (continued)

Attribute	Use
Response business object	Contains the response message, if the interaction is synchronous.

For more information on how to create this business-object structure, see “Creating business object definitions for pre-4.2.2 ICS over HTTP” on page 87.

Receiving documents from pre-4.2.2 ICS through HTTP

This section provides the following information to describe how to receive documents from Business Integration Connect from a pre-4.2.2 InterChange Server over the HTTP transport protocol:

- “Components required for receiving”
- “Setting up the environment for HTTP with pre-4.2.2 ICS” on page 84
- “Creating business object definitions for receiving documents” on page 86

The document that Business Integration Connect receives from InterChange Server as been initiated by request processing within InterChange Server.

Components required for receiving

Business Integration Connect can receive documents from the following pre-4.2.2 versions of InterChange Server over the HTTP transport protocol:

- Version 4.1.1
- Version 4.2.0
- Version 4.2.1

For Business Integration Connect to receive a document from pre-4.2.2 InterChange Server using the HTTP transport protocol requires that these two components be configured. Table 45 on page 68 summarizes these configuration steps. In addition, to receive a document from InterChange Server over the HTTP protocol, you use the ICS-compatible components listed in Table 53..

Table 53. Components required to receive documents to pre-4.2.2 InterChange Server through HTTP

Component	Description	Notes and restrictions
WebSphere Business Integration Adapter for XML (Adapter for XML)	This adapter his adapter allows InterChange Server to exchange business objects with applications that receive data in the form of HTTP messages. The Adapter for XML and Business Integration Connect communicate through a URL address.	The Adapter for XML is <i>not</i> shipped with Business Integration Connect. You must use version 3.1.x or higher of this adapter. Note: The adapter can <i>only</i> be used with version 4.2.2 WebSphere InterChange Server.
The HTTP or HTTPS protocol handler	This protocol handler works with the Adapter for XML to send information stream to and receive them from the URL.	This protocol handler is provided with Business Integration Connect. For more information, see “Deploying the HTTP protocol handler” on page 85.
A payload data handler	This data handler converts the document payload between its document format (usually XML) and its business-object representation.	This data handler is required and must support the MIME type of your payload document.

Table 53. Components required to receive documents to pre-4.2.2 InterChange Server through HTTP (continued)

Component	Description	Notes and restrictions
Attachment data handler	This data handler converts documents that contain attachments between their document format and their business-object representation.	This data handler is required only if your documents include attachments. For more information, see "Handling documents with attachments" on page 49.

Figure 13 provides an overview of how Business Integration Connect receives documents from a pre-4.2.2 InterChange Server over the HTTP transport protocol.

Note: All references to the HTTP protocol handler apply to the HTTPS protocol handler as well.

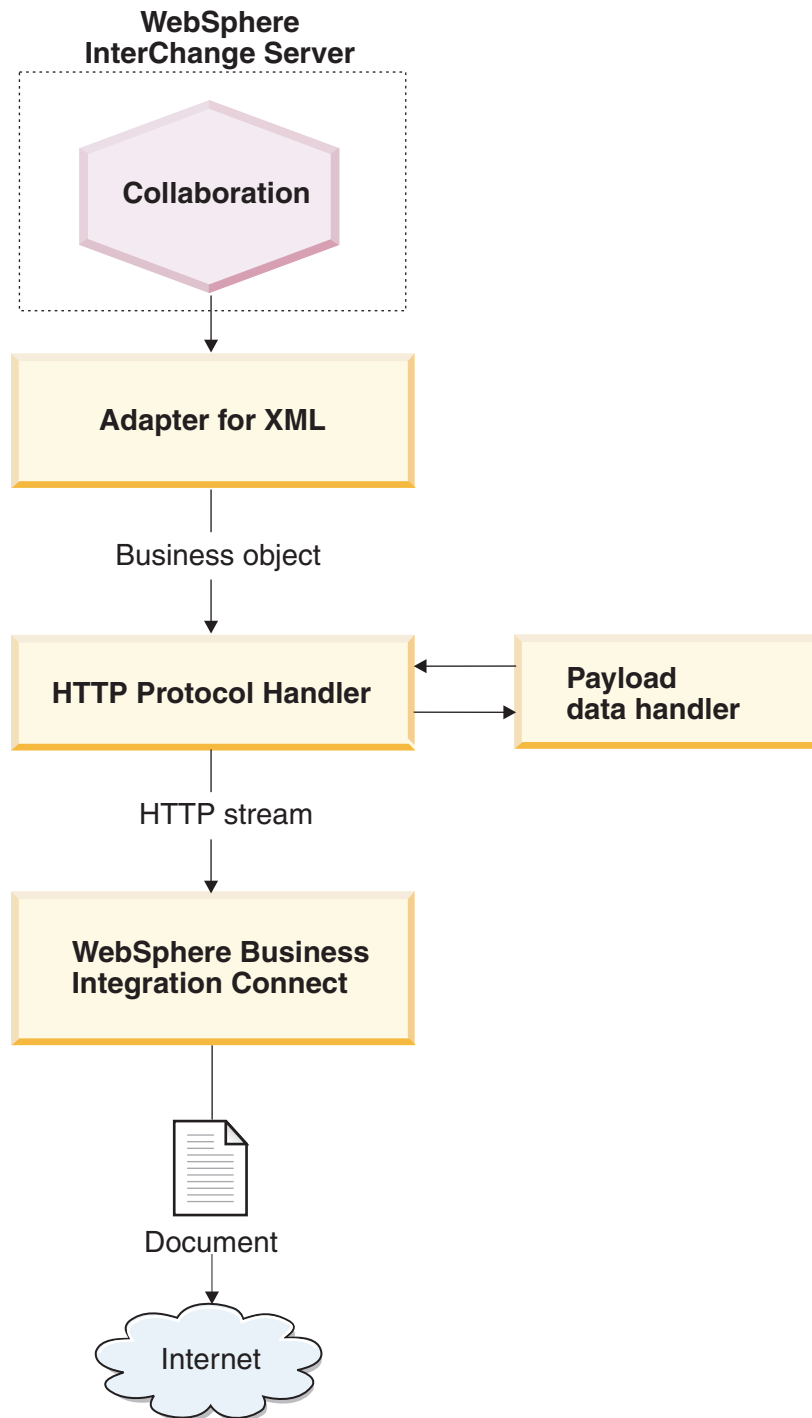


Figure 13. Message flow from a collaboration to Business Integration Connect through the HTTP transport protocol

The following steps describe how Business Integration Connect participates in request processing by receiving a document initiated by a collaboration within InterChange Server:

1. The collaboration within InterChange Server makes a service call to the Adapter for XML, sending it a top-level business object that includes request and response child objects.

The request child object contains application-specific information pointing to a dynamic meta-object that contains the custom HTTP headers, which Business Integration Connect expects.

2. The Adapter for XML invokes the HTTP protocol handler.
3. The HTTP protocol handler uses a data handler to convert the business object that the collaboration has sent into an HTTP stream.
The protocol handler reads the MIME type and URL from the top-level business object to determine the data handler to use and the address of the recipient.
4. From the top-level business object, the HTTP protocol handler obtains the first populated business object. This is the request business object.
The HTTP protocol handler calls the data handler to convert the business object to an HTTP stream.

Note: If your documents have attachments, install the Attachment data handler and then configure the Adapter for XML to call it to convert the require business object to a document with attachments. For more information, see “Handling documents with attachments” on page 49.

5. The HTTP protocol handler determines, from the request business object, the name of the dynamic meta-object.
The HTTP protocol handler searches the application-specific information of the request business object for the `cw_mo_conn` tag, which identifies the attribute that corresponds to the dynamic meta-object. If you are using Backend Integration packaging for your documents, you can specify the custom HTTP header information in this dynamic meta-object.
6. The HTTP protocol handler searches the dynamic meta-object for the `HTTPProperties` attribute.
If this attribute is populated, the protocol handler sets the transport-level headers in the request message. Within the `HTTPProperties` attribute, you can also specify the content-type standard HTTP header. For more information, see “Creating HTTP transport-level header information for v4.2.2 ICS” on page 106.
7. The HTTP protocol handler creates an HTTP stream, using the string returned by the data handler. It also sets any custom header information, as defined in the dynamic meta-object.
8. The HTTP protocol handler sends the resulting request message as a stream to the specified URL.
Business Integration Connect listens on this URL, which is configured as its target.
9. Business Integration Connect responds with an HTTP 200 OK.
If the `ReturnBusObjResponse` connector property (of the Adapter for XML) is true, the invocation is synchronous. The protocol handler converts the response message into a response business object and returns it to the Adapter for XML. The adapter sets the business object in the top-level business object. The top-level business object is then returned to the collaboration within InterChange Server.

Setting up the environment for HTTP with pre-4.2.2 ICS

Because the receiving of documents from InterChange Server involves the use of ICS-compatible components, you must perform the setup and configuration tasks described in Table 54.. For information on how to configure Business Integration Connect for communication with a pre-4.2.2 InterChange Server over HTTP, see “Providing support for outgoing documents” on page 41.

Table 54. Setting up the environment for sending documents

Step	For more information
1. Deploy the HTTP protocol handler.	"Deploying the HTTP protocol handler"
2. Configure the WebSphere Business Integration Adapter for XML.	"Configuring the Adapter for XML"

Note: If your documents contain attachments, you must also install and configure the Attachment data handler, as described in "Handling documents with attachments" on page 49.

Deploying the HTTP protocol handler: Business Integration Connect provides a custom HTTP protocol handler to send and receive messages to Business Integration Connect. This HTTP protocol handler is available on the Business Integration Connect installation medium in the following file:

Integration/WBI/WICS/WBICServlet/bcgwbiprotocol.jar

This custom protocol handler can be plugged into the Adapter for XML version 3.1.x or higher. For a list of supported InterChange Server versions and platforms, refer to the *Adapter for XML User Guide* for the version of the adapter you are using.

To deploy the HTTP protocol handler to the Adapter for XML, you must let the Adapter for XML know the location of the HTTP protocol handler, so that it can load it at run-time. To specify the location of the HTTP protocol handler, follow these steps:

1. Edit the startup script for the Adapter for XML, `start_xml.bat`, which is located in the following subdirectory of the product directory in which your WebSphere Business Integration Adapters are installed:
connectors/xml
2. In this startup script, add the jar file for the custom HTTP protocol handler, `bcgwbiprotocol.jar`, to the list of jar files in the CLASSPATH of the Adapter for XML.

Configuring the Adapter for XML: The Adapter for XML is the ICS-compatible component that allows Business Integration Connect to exchange documents with InterChange Server in the form of HTTP messages. It supports the request processing interaction with InterChange Server as follows:

- It receives the business object from InterChange Server.
- It converts the business object to an HTTP stream, using the HTTP protocol handler.
- It sends the HTTP stream to a specified URL, where it can be retrieved by Business Integration Connect.

Note: The event notification feature of this adapter is not used. To send HTTP messages from Business Integration Connect to InterChange Server, use the WebSphere Business Integration Connect Servlet, as described in "Sending documents to pre-4.2.2 ICS through HTTP" on page 67

Important: WebSphere Business Integration Connect does *not* include the WebSphere Business Integration Adapter for XML. You must obtain this product separately and install it according to the instructions in its

Adapter for XML User Guide. Refer to the adapter documentation to ensure that the version of the adapter is compatible with the version of InterChange Server you are using.

When you have configured the Adapter for XML to communicate with InterChange Server, follow the steps in these sections to configure this adapter to accept HTTP messages from Business Integration Connect.

Specifying the payload data handler: As Figure 13 on page 83 shows, the Adapter for XML's protocol handler uses a data handler to convert into the appropriate HTTP streams the business objects it receives from InterChange Server.

Note: The data handler that the Adapter for HTTP calls converts the payload of the document. If your document is wrapped in an XML transport envelope (it contains attachments or the Envelope Flag is Yes), configure the Attachment data handler as the payload data handler. For more information, see "Handling documents with attachments" on page 49.

To indicate which data handler to use to convert the payload, you must take the steps listed in "Business object conversion" on page 46. In addition, you must configure the Adapter for XML to use this payload data handler. In Connector Configurator, set the `DataHandlerConfigM0` connector configuration property to specify the top-level data-handler meta-object that the Adapter for XML uses to identify data handlers. Make sure you include the name of the top-level data-handler meta-object in the list of supported business objects for the adapter.

Configuring the protocol-handler package name: The Adapter for XML uses the `JavaProtocolHandlerPkgs` connector configuration property to identify the name of the Java Protocol Handler packages. For integration with Business Integration Connect, make sure that the `JavaProtocolHandlerPkgs` property is set to the package name for the Business Integration Connect-provided HTTP protocol handler:

```
com.ibm.bcg.integration.wbi.utils.protocolhandlers
```

Specifying support for a response business object: The Adapter for XML uses the `ReturnBusObjResponse` connector configuration property to indicate whether to return a response business object. A response business object is returned *only* if the interaction is synchronous. By default, the `ReturnBusObjResponse` connector configuration property is set to false. To configure the Adapter for XML to return a response business object, set the `ReturnBusObjResponse` connector configuration property to true.

Note: If Business Integration Connect supports synchronous interactions for the packaging and business protocol that the Community Manager is using, set the `ReturnBusObjResponse` connector configuration property to true and provide the response business object in your top-level business object.

To set connector configuration properties, use the Connector Configurator tool, which is included as part of the release for your WebSphere Business Integration Adapter for XML. Within Connector Configurator, the `ReturnBusObjResponse` property should appear in the Connector-specific tab of the connector properties.

Creating business object definitions for receiving documents

The WebSphere Business Integration Adapter for XML receives information from InterChange Server in the form of a payload business object. For the Adapter for XML, the payload business object is represented as a hierarchy of business objects.

The Adapter for XML creates this business-object hierarchy when it receives a Business Integration Connect document. Therefore, you must create business object definitions to represent this hierarchy.

Because the Adapter for XML participants *only* in request processing with InterChange Server, the request and response attributes of the top-level business object are interpreted as shown in Table 55.

Table 55. Request and response business objects in request processing

Attribute	Use
Request business object	Contains the request information from InterChange Server; the protocol handler and the data handler convert and send this information to the URL on which Business Integration Connect listens.
Response business object	Contains the response information from Business Integration Connect, if the interaction is synchronous.

For more information on how to create this business-object structure, see “Creating business object definitions for pre-4.2.2 ICS over HTTP.”

Creating business object definitions for pre-4.2.2 ICS over HTTP

The Connect Servlet sends your document to InterChange Server in the form of a payload business object. The Adapter for XML receives your message from InterChange Server in this same form. Both these components invoke the payload data handler to handle this business object when it receives or sends a Business Integration Connect document, as follows:

- For request processing, the payload data handler converts the request business object to its corresponding HTTP stream.
- For event notification, the data handler converts the HTTP stream to an event business object.

Therefore, you must create the business object definitions shown in Table 56. to represent the payload business-object structure that the Adapter for XML and Connect Servlet expect.

Table 56. Business object definitions for HTTP transport protocol

Condition	Business object definition	For more information
If you are using either None or Backend Integration packaging for your document <i>and</i> your documents do <i>not</i> have attachments	Hierarchy of business objects for the payload business object: <ul style="list-style-type: none"> • Top-level business object • Request business object • Response business object (only if a response is expected) 	“Creating the payload business-object structure for pre-4.2.2 ICS over HTTP” on page 88

Table 56. Business object definitions for HTTP transport protocol (continued)

Condition	Business object definition	For more information
If you are using Backend Integration packaging for your document	Add to the payload business object the business objects to hold the transport-level header information: <ul style="list-style-type: none"> • Dynamic meta-object • HTTP-properties business object 	“Creating HTTP transport-level header information for pre-4.2.2 InterChange Server” on page 92
If the document includes attachments (Backend Integration packaging is required)	You must also create additional business objects to represent the attachments.	“Creating attachment-related business object definitions” on page 60

Note: If you are defining business objects for cXML documents, see “Creating business objects for cXML” on page 92.

Creating the payload business-object structure for pre-4.2.2 ICS over HTTP

Both the Wrapper data handler (for sending documents), as well as the Adapter for XML and the HTTP protocol handler (for receiving documents) expect the same business-object structure for the payload business object. This business-object structure consists of the following business objects:

- A top-level business object
- A request business object
- A response business object (optional)

Figure 14 shows a sample business-object structure for the payload business object definition for use with a pre-4.2.2 InterChange Server over the HTTP transport protocol.

Note: For a detailed description of this business-object structure, refer to the *Adapter for XML User Guide*.

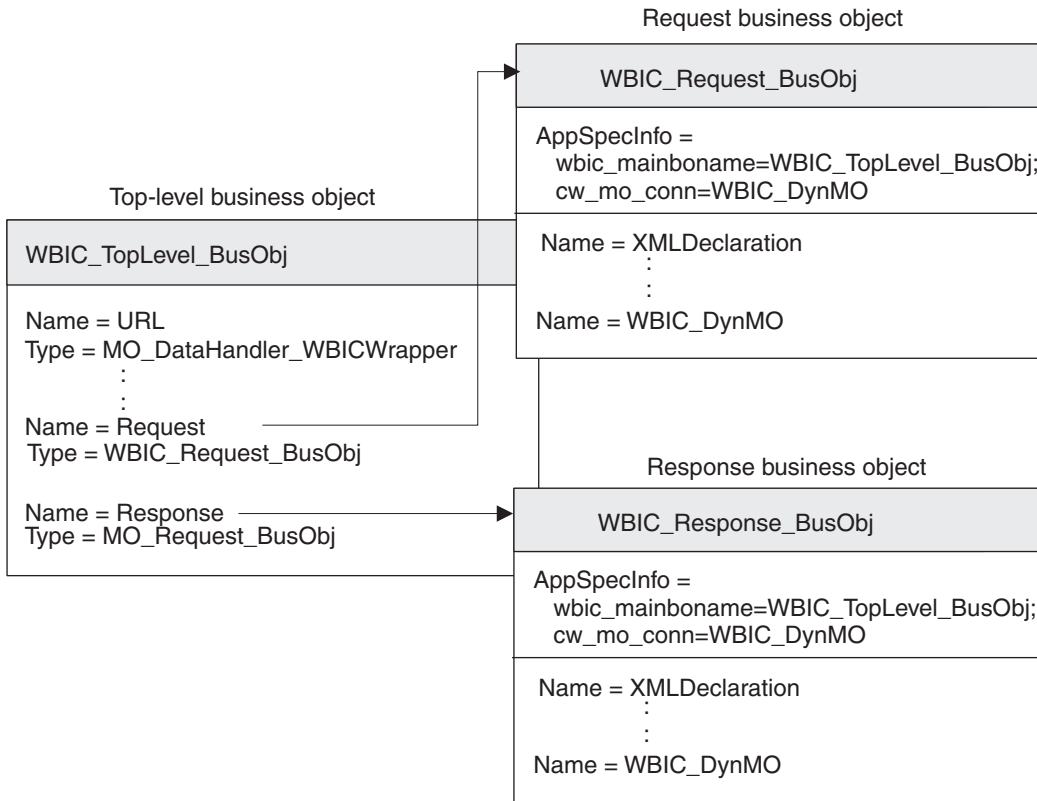


Figure 14. Business-object structure for the HTTP payload business object for pre-4.2.2 InterChange Server

Top-level business object: The top-level business object is a wrapper for the request and response business objects. You must create a business object definition for this business object. Table 57 summarizes the attributes of this top-level business object definition.

Table 57. Attributes of the top-level business object

Attribute	Attribute type	Description
URL	String	Destination of the data in the business object. Important: This attribute is <i>not</i> used by the Wrapper data handler. However, it is used by the Adapter for XML.
MimeType	String	Defines the content type and format of the data that is being passed to the URL. Important: This attribute is <i>not</i> used by the Wrapper data handler. However, it is used by the Adapter for XML.
BOPrefix	String	Used to determine which data handler to call. Important: This attribute is <i>not</i> used by the Wrapper data handler.

Table 57. Attributes of the top-level business object (continued)

Attribute	Attribute type	Description
Response	Business object	Child business object that represents the response message (if you are expecting a response). The purpose of this business object depends on whether it participates in request processing or event notification. For more information on the structure of this business object, see "Response business object" on page 91.
Request	Business object	Child business object that represents the request message. The purpose of this business object depends on whether it participates in request processing or event notification. For more information on the structure of this business object, see "Request business object."

Note: If you are using the Attachment data handler to process attachments, you must modify your request business object to hold the attachments, as described in "Creating attachment-related business object definitions" on page 60.

For a complete description of the structure of the top-level business object, see the *Adapter for XML User Guide*.

Request business object: The request business object contains the data to be passed to the URL. It contains attributes for the various XML tags in the request message. The purpose of this request business object depends on which InterChange Server task it is participating in, as follows:

- For event notification (sending a document to InterChange Server), the request business object contains the request message from Business Integration Connect, which is the event to be sent to InterChange Server.
For more information, see "Creating business object definitions for sending documents" on page 80.
- For request processing (receiving a document from InterChange Server), the request business object contains the request that InterChange Server is making to Business Integration Connect.
For more information, see "Creating business object definitions for receiving documents" on page 86.

Note: This business-object structure identifies its two child business objects as its request and response business objects. However, this structure is used in *both* request processing and event notification,

For the basic description of the request business object's structure, refer to the *Adapter for XML User Guide*. For use with Business Integration Connect, there are two customizations you must make to the structure of the request business object definition:

- If the document that Business Integration Connect sends to InterChange Server uses Backend Integration packaging, you must add to the request business object definition a special attribute to identify the dynamic meta-object.

This attribute provides configuration information for the headers of the message. For more information, see “Creating HTTP transport-level header information for pre-4.2.2 InterChange Server” on page 92.

- To the business-object-level application-specific information of the request business object definition, add the tags shown in Table 58.

Table 58. Tags in application-specific information of request business object

Application-specific-information tag	Description	Required?
wbic_mainboname	Gives the name of the top-level business object	Yes
cw_mo_conn	Specifies the dynamic meta-object, which contains the HTTP transport-level header fields. For more information, see “Creating HTTP transport-level header information for pre-4.2.2 InterChange Server” on page 92.	No (only required if you are using Backend Integration packaging)

Response business object: The response business object contains the data to be received from the URL. It contains attributes for the various XML tags in the response message. The purpose of this response business object depends on which InterChange Server task it is participating in, as follows:

- For event notification, the response business object contains the response from the collaboration. For more information, see “Creating business object definitions for sending documents” on page 80.
- For request processing, the response business object contains the information from the URL in response to the request that InterChange Server sent. For more information, see “Creating business object definitions for receiving documents” on page 86.

Regardless of whether the response is part of event notification or request processing, a response business object is sent *only* if the exchange between Business Integration Connect and InterChange Server is *synchronous* and a business response is expected in response to your request. If this is the case, you must take the following additional steps:

- In the top-level business object, add the `wbic_type` tag to the attribute-level application-specific information for the attribute that corresponds to the response business object.

This tag has the following syntax:

```
wbic_type=reply
```

- Decide whether to add the `wbic_response_mime` business-object level application-specific information. This application-specific information is optional. It specifies the MIME type for the data handler to be used for the response business object.

If this tag is *not* specified, the Wrapper data handler uses the child meta-object indicated by the `wbic_response_mime` attribute (in the top-level business object) to determine the data handler to use for the response.

Note: The response business object does *not* include an attribute for the dynamic meta-object.

If the exchange between Business Integration Connect and InterChange Server is *asynchronous*, Business Integration Connect does *not* expect a response, so you do not need to create a response business object.

Creating business objects for cXML: For cXML documents, you can use the XML Object Discovery Agent (ODA) to create the business objects. The XML ODA can consume the cXML DTD. Note, however, that the XML ODA does not support ENTITY. Therefore, before running the cXML DTD with the XML ODA, you need to remove ENTITY from the DTD.

When generating business objects using the XML ODA, you can select the cXML tag as your root element. This might result in a large business object, capturing the entire cXML DTD, however. If you want to create a smaller business object, you can select a different tag as your root element, which will require that you write a custom name handler for the Data Handler for XML. The data handler will invoke this name handler for the top-level business object name resolution. Refer to the Data Handler for XML documentation for information on writing custom name handlers.

Creating HTTP transport-level header information for pre-4.2.2 InterChange Server

If you are sending documents with Backend Integration packaging over the HTTP transport protocol, your request business object needs to contain custom transport-level header information. Both the Wrapper data handler and the Adapter for XML expect this custom header information to be in a **dynamic meta-object**.

Figure 15 shows the business-object structure for a request business object that represents a Business Integration Connect document with Backend Integration packaging over the HTTP transport protocol.

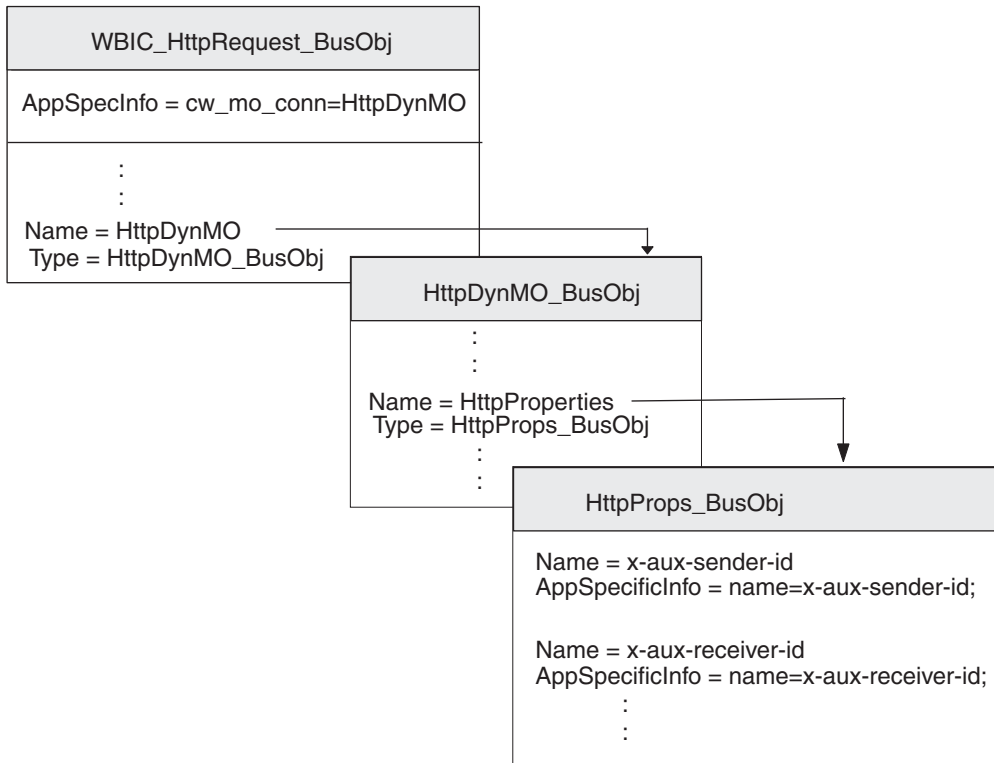


Figure 15. Relationship of the request business object to the HTTP dynamic meta-object

Make sure your business-object structure includes a dynamic meta-object by taking the following steps:

1. Create a business object definition to hold the HTTP properties required by the Backend Integration packaging.
2. Create a business object definition for the dynamic meta-object.
3. Modify the business object definition for your request business object to include an attribute for the dynamic meta-object.

Each of these steps is described in the sections below.

Creating the HTTP-properties business object: An **HTTP-properties business object** contains HTTP properties required by the Backend Integration packaging. It can also contain the Content-Type attribute, which specifies the content-type header to set in the request message, and the content-length attribute, which specifies the length of the message, in bytes. Table 4 on page 11 describes each of the valid transport-header fields.

To create an HTTP-properties business object definition, take the following steps:

1. Create an attribute within the business object definition for each of the transport-header fields.

All attributes should have an attribute type of String. You can name the attribute with the exact name of the HTTP property (as listed in the Header field column of Table 4 on page 11).

Note: The only exception to the HTTP property names is that the content-type field should have an attribute named Content_Type.

2. For each of the attributes in the HTTP-properties business object, add application-specific information to identify the purpose of the associated attribute.

This attribute-level application-specific information has the following format:

```
name=HTTPproperty
```

where *HTTPproperty* is one of the values in the Header field column of Table 4 on page 11.

In Figure 15 on page 93, the `HttpProps_BusObj` business object definition contains attributes for the various transport-header fields. These attributes all have attribute-level application-specific information to specify the name of the related protocol header. For example, the `x-aux-sender-id` attribute has the application-specific information set as follows:

```
name=x-aux-sender-id
```

Creating the HTTP dynamic meta-object: The **dynamic meta-object** contains a child business object with configuration information for the HTTP header information. You must make sure that your business-object structure includes a dynamic meta-object. The business object definition for the dynamic meta-object *must* include an attribute named `HttpProperties`, whose attribute type is the business object definition for the HTTP-properties business object (see “Creating the HTTP-properties business object” on page 93).

For example, in Figure 15 on page 93, the `HttpDynM0_BusObj` business object definition contains attributes the `HttpProperties` attribute, whose attribute type is `HttpProps_BusObj`.

Modifying the request business object definition: The request business object definition represents the information being requested from Business Integration Connect. For information on how to create the request business object, see “Request business object” on page 90. To incorporate the dynamic meta-object into your payload business-object structure, you must make the following modifications to your request business object definition:

1. Add an attribute to your request business object definition to hold your dynamic child meta-object.

The attribute type for this attribute is the business object definition for the dynamic meta-object (see “Creating the HTTP dynamic meta-object”).

2. Add the `cw_mo_conn` tag to the business-object-level application-specific information of your request business object definition to identify the attribute that contains the dynamic meta-object.

The `cw_mo_conn` tag has the following format:

```
cw_mo_conn=dynamicMetaObjAttr
```

where *dynamicMetaObjAttr* is the name of the attribute in the request business object that holds the dynamic meta-object.

For example, in Figure 15 on page 93, an attribute named `HttpDynM0` has been added to the request business object definition, `WBIC_HttpRequest_BusObj`. This attribute contains the dynamic meta-object, which is a child business object of type `HttpDynM0_BusObj`. In addition, the application-specific information of the request business object has been modified to include the following `cw_mo_conn` tag to identify this dynamic meta-object:

```
cw_mo_conn=HttpDynM0
```

Creating pre-4.2.2 ICS artifacts for HTTP

To configure a pre-4.2.2 InterChange Server for communication with Business Integration Connect over the HTTP transport protocol, you must create the InterChange Server artifacts shown in Table 59.

Table 59. Artifacts for communicating with pre-4.2.2 ICS over the HTTP transport protocol

ICS artifact	Purpose	For more information
Business object definitions	Represent the document	“Creating business object definitions for pre-4.2.2 ICS over HTTP” on page 87
Connector object (needed for request processing only)	Represent the Adapter for XML at run-time	“Creating the XML connector object”
Collaboration template and collaboration object	Represents the business process that InterChange Server uses to process the document	“Binding collaborations to communicate with Adapter for XML”

Creating the XML connector object

To support request processing with pre-4.2.2 InterChange Server over HTTP transport protocol, you use the Adapter for XML to send a document to InterChange Server. To obtain an instance of the Adapter for XML at run-time, take the following steps within System Manager:

1. Create the connector objects:
 - Create a connector object to represent an instance of the Adapter for XML.

Note: In the Supported Business Objects tab of Connector Configurator, make sure that you specify all business object definitions you created for use with the Adapter for XML. For a description of these business object definitions, see “Creating business object definitions for pre-4.2.2 ICS over HTTP” on page 87.

- If required by your collaboration, create a connector object for the Port Connector.
2. Configure the connector objects.

For information on how to configure your Adapter for XML for use with Business Integration Connect, see “Configuring the Adapter for XML” on page 85.

Binding collaborations to communicate with Adapter for XML

As described in “Creating the collaborations” on page 48, a collaboration object must exist at run-time for InterChange Server to know where to receive and send business objects. When you create the collaboration object for the collaboration that sends information to Business Integration Connect, you bind its ports. For request processing, set the “to” collaboration port, which uses the Adapter for XML to send requests to Business Integration Connect, to the connector object you created for the Adapter for XMP; that is, the Adapter for XMP is the *destination* adapter.

Using HTTP transport protocol with v4.2.2 ICS

WebSphere Business Integration Connect can send and receive documents with a version 4.2.2 WebSphere InterChange Server (ICS) over the HTTP transport protocol.

Notes:

1. To send and receive documents between WebSphere Business Integration Connect and a *pre-4.2.2* WebSphere InterChange Server over the HTTP transport protocol, see “Using HTTP transport protocol with pre-4.2.2 ICS” on page 67.
2. If you are exchanging SOAP documents over the HTTP transport protocol, see “Sending SOAP documents over HTTP/S” on page 110.

This section provides the following information on how to configure a v4.2.2 InterChange Server and the appropriate ICS-compatible components for use with Business Integration Connect over HTTP:

- “Components required for documents to v4.2.2 ICS over HTTP transport”
- “Setting up the environment for HTTP transport with v4.2.2 ICS” on page 99
- “Creating business object definitions for v4.2.2 ICS over HTTP” on page 102
- “Creating v4.2.2 ICS artifacts for HTTP” on page 109

Components required for documents to v4.2.2 ICS over HTTP transport

For Business Integration Connect to communicate with a v4.2.2 of InterChange Server using the HTTP transport protocol requires that these two components be configured. Table 60 summarizes these configuration steps.

Table 60. Configuring Business Integration Connect and InterChange Server

Component	Version	For more information
WebSphere Business Integration Connect	4.2.2	“Configuring for outgoing documents over HTTP transport protocol” on page 41
		“Configuring for incoming documents over HTTP transport protocol” on page 43
WebSphere InterChange Server	4.2.2	“Creating v4.2.2 ICS artifacts for HTTP” on page 109

In addition, to send or receive a document between Business Integration Connect and a v4.2.2 of InterChange Server using the HTTP transport protocol, you use the ICS-compatible components listed in Table 61.

Table 61. Components required to transfer documents with v4.2.2 InterChange Server through HTTP

Component	Description	Notes and restrictions
WebSphere Business Integration Adapter for HTTP	This adapter allows InterChange Server to exchange business objects with applications that send or receive data in the form of HTTP streams.	This adapter <i>cannot</i> be used with WebSphere InterChange Server <i>before</i> version 4.2.2.
(Adapter for HTTP) A payload data handler	This data handler converts the document payload between its document format (usually XML) and its business-object representation.	This data handler is required and must support the MIME type of your payload document.
Attachment data handler	This data handler handles attachment documents for your document message.	This data handler is required <i>only</i> if your documents include attachments.

The following sections describe how the components in Table 61 on page 96 work together to send and receive documents between Business Integration Connect and v4.2.2 InterChange Server over the HTTP transport protocol.

Sending documents to v4.2.2 ICS through HTTP

For Business Integration Connect to send a document to v4.2.2 InterChange Server using the HTTP transport protocol, you use the Adapter for HTTP to retrieve the document that Business Integration Connect has sent as an HTTP stream. The adapter then routes the document to InterChange Server. Figure 16 provides an overview of how Business Integration Connect sends documents to v4.2.2 InterChange Server over the HTTP transport protocol.

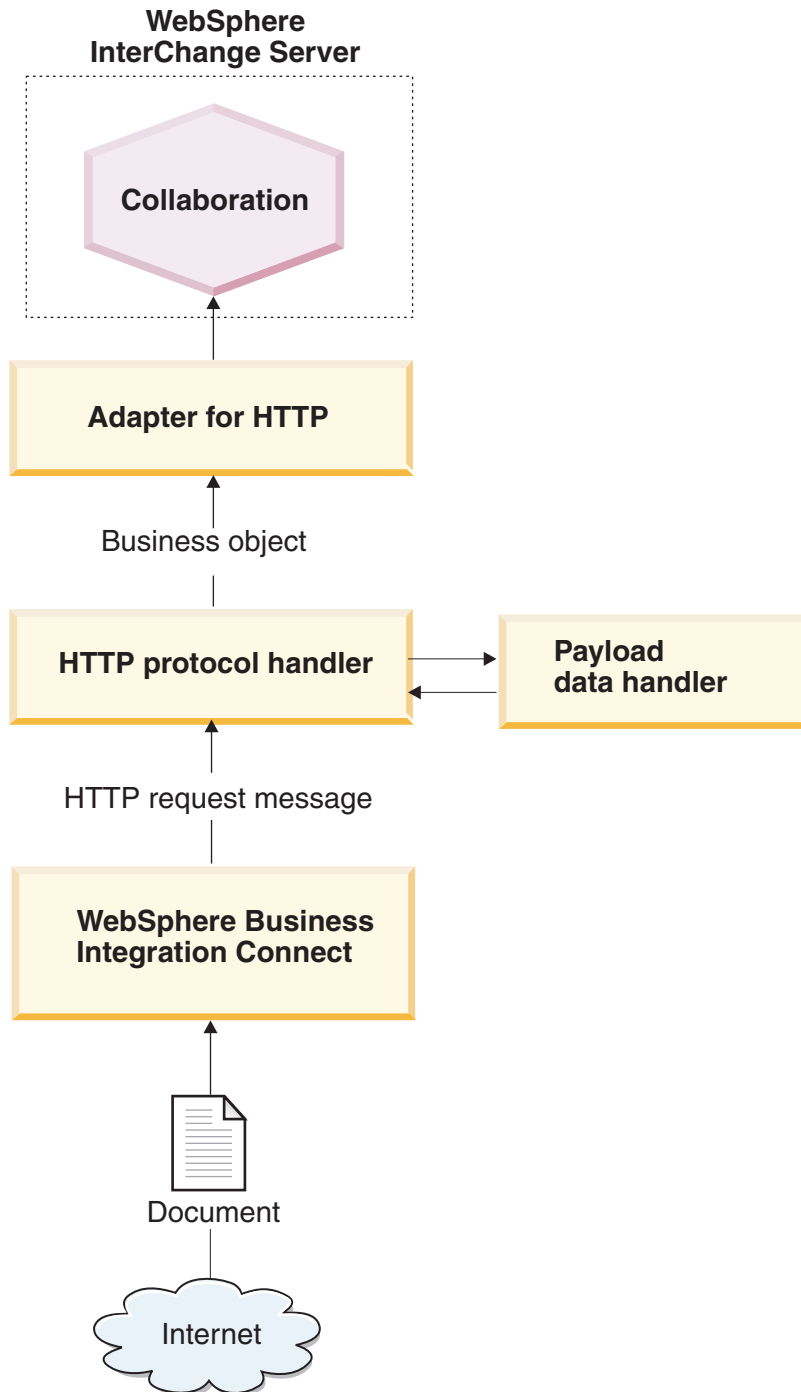


Figure 16. Message flow from Business Integration Connect to a collaboration through the HTTP transport protocol

Receiving documents from v4.2.2 ICS through HTTP

For Business Integration Connect to receive a document from v4.2.2 InterChange Server using the HTTP transport protocol, you use the Adapter for HTTP, which sends the message it receives from InterChange Server as an HTTP stream for Business Integration Connect to retrieve. Figure 17 provides an overview of how

Business Integration Connect receives documents from v4.2.2 InterChange Server over the HTTP transport protocol.

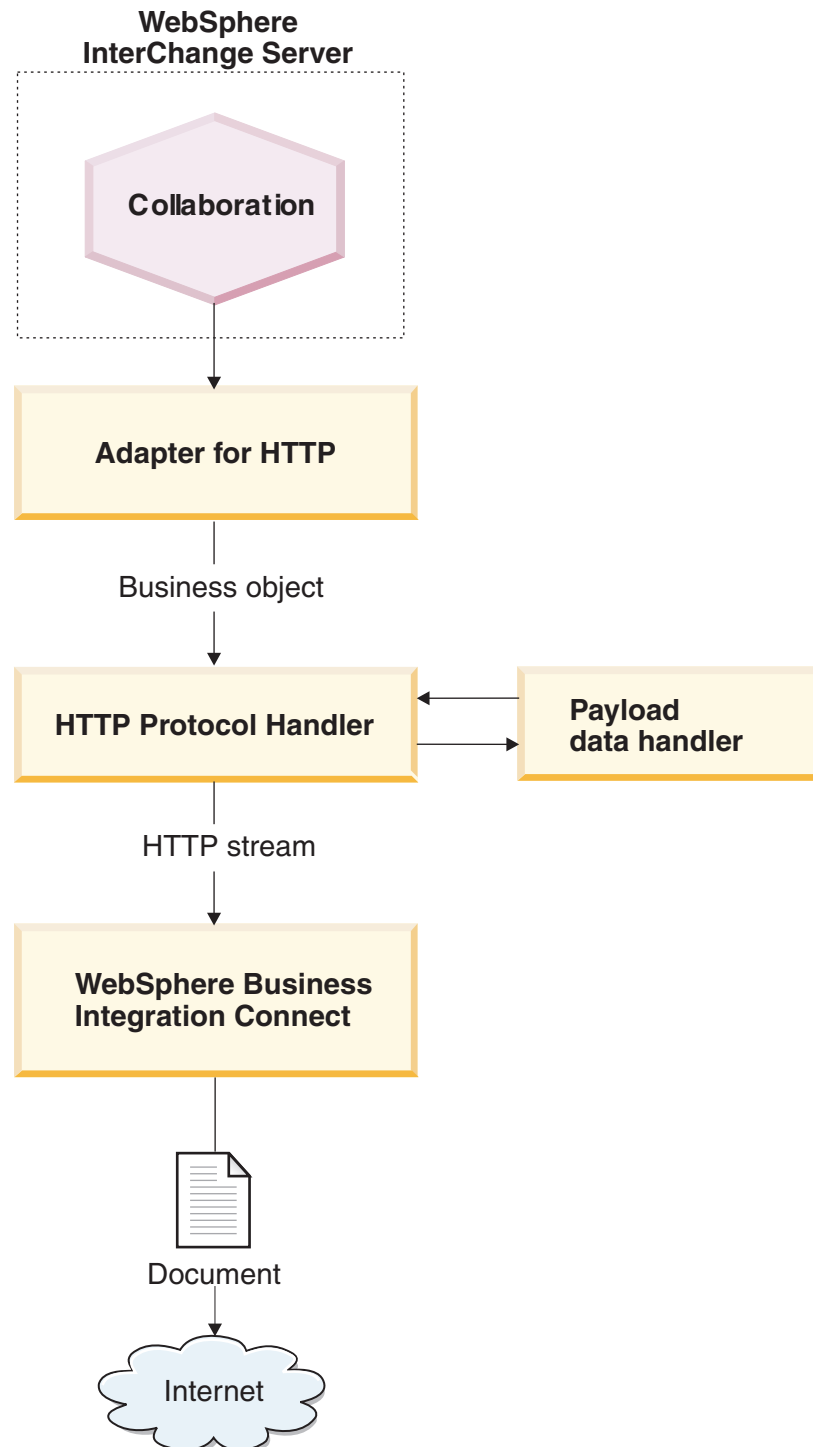


Figure 17. Message flow from a collaboration to Business Integration Connect through the HTTP transport protocol

Setting up the environment for HTTP transport with v4.2.2 ICS

Because the sending and receiving of documents to and from InterChange Server involves ICS-compatible components, you must perform the setup and

configuration tasks on the Adapter for HTTP. For information on how to configure Business Integration Connect for use with InterChange Server over HTTP, see “Configuring Business Integration Connect for InterChange Server” on page 41.

The Adapter for HTTP is the ICS-compatible component that allows Business Integration Connect to exchange documents with v4.2.2 InterChange Server in the form of HTTP messages. It supports the following interactions with InterChange Server:

- For request processing, it receives the request business object from InterChange Server, converts it to an HTTP stream, and sends it to the specified URL, where it can be received by Business Integration Connect.
- For event notification, it listens at a specified URL, where Business Integration Connect sends documents. When it receives a document, it converts it to an event business object (using a data handler) and sends it to InterChange Server.

Important: WebSphere Business Integration Connect does *not* include the WebSphere Business Integration Adapter for HTTP. You must obtain this product separately and install it according to the instructions in its *Adapter for HTTP User Guide*. Refer to the adapter documentation to ensure that the version of the adapter is compatible with the version of InterChange Server you are using.

When you have configured the Adapter for HTTP to communicate with InterChange Server, follow the steps in these sections to configure this adapter for listen for HTTP messages from Business Integration Connect:

Specifying the payload data handler

As Figure 17 on page 99 shows, the Adapter for HTTP uses a data handler to convert the business objects it receives from InterChange Server into the appropriate HTTP streams.

Note: The data handler that the Adapter for HTTP calls converts the payload of the document. If your document is wrapped in an XML transport envelope (it contains attachments or the Envelope Flag is Yes), configure the Attachment data handler as the payload data handler. For more information, see “Handling documents with attachments” on page 49.

To indicate which data handler to use to convert the payload, you must take the steps listed in “Business object conversion” on page 46. In addition, you must configure the Adapter for HTTP to use this payload data handler. You can set the payload data handler in either of the following ways:

- In Connector Configurator, set the `DataHandlerMetaObjectName` connector configuration property to specify the name of the top-level data-handler meta-object that the Adapter for HTTP uses to identify data handlers. Make sure you include the top-level data-handler meta-object in the list of supported business objects for the adapter.
- In the top-level business object, use the `MimeType` attribute to hold the MIME type to identify the payload data handler. For more information on this business object, see “Top-level business object” on page 103.

Configuring the protocol-handler package name

The Adapter for HTTP uses the `JavaProtocolHandlerPackages` connector configuration property to identify the name of the Java Protocol Handler packages. For integration with Business Integration Connect, make sure that the `JavaProtocolHandlerPackage` property is set to its default value:

Configuring the HTTP protocol listener

The Adapter for HTTP supports hierarchical configuration properties to obtain the information it needs to configure its protocol listeners. The top-level configuration property is called `ProtocolListenerFramework`. Within this top-level property are several levels of subproperties. To configure the protocol handlers for use with the Adapter for HTTP, make sure that the properties are configured in the `ProtocolListener` property, as described in the following steps:

1. Configure a protocol listener with subproperties under the following configuration property:

```
ProtocolListenerFramework
  ProtocolListeners
    HttpListener1
```

To configure your protocol listener, set the subproperties listed in Table 62.

Table 62. Configuring the protocol listener

Property	Description	Value
Protocol	Type of protocol listener: <ul style="list-style-type: none"> • HTTP • HTTPS 	http or https
Host	IP address on which the protocol listener listens	IP address of the local machine on which WebSphere Business Integration Connect is running
Port	Port on which the protocol listener listens for requests	8080

2. Configure the URL configurations that the protocol listener supports with subproperties under the following configuration property:

```
ProtocolListenerFramework
  ProtocolListeners
    HttpListener1
      URLsConfiguration
        URL1
```

Set the `ContextPath` property to the URI for the HTTP requests that the protocol listener receives.

Note: This directory must be the same one that the Business Integration Connect gateway specifies as its Target URI. For more information, see “Configuring for outgoing documents over HTTP transport protocol” on page 41.

3. If your document contains attachments, you must configure a transformation for the protocol listener by setting subproperties of the following configuration property:

```
ProtocolListenerFramework
  ProtocolListeners
    HttpListener1
      URLsConfiguration
        URL1
          TransformationRules
            TransformationRule1
```

To configure the attachment transformation for your protocol listener, set the subproperties listed in Table 63. You need one transformation rule for each instance of the Attachment data handler you are using. For more information on the Attachment data handler, see “Handling documents with attachments” on page 49

on page 49.

Table 63. Configuring the attachment transformation for the protocol listener

Property	Description	Value
ContentType	Content type of the data to be transformed with a data handler	Content type associated with the attachment data
MimeType	MIME type to use to identify the data handler to call	MIME type associated with the instance of the Attachment data handler
Charset	Character set to use when transforming data of the specified content type	Character set for the attachment data

For more information on these properties, see the *Adapter for HTTP User Guide*.

Creating business object definitions for v4.2.2 ICS over HTTP

The Adapter for HTTP sends and receives your document to InterChange Server in the form of a payload business object. The Adapter for HTTP invokes the payload data handler to handle this business object when it receives or sends a Business Integration Connect document, as follows:

- For request processing, the payload data handler converts the request business object to its corresponding HTTP stream.
- For event notification, the data handler converts the HTTP stream to an event business object.

Therefore, you must create the business object definitions shown in Table 64 to represent the payload business-object structure that the Adapter for HTTP expects.

Table 64. Business object definitions for the Adapter for HTTP

Condition	Business object definition	For more information
If you are using None or Backend Integration packaging for your message and your documents do <i>not</i> have attachments	Payload business object: <ul style="list-style-type: none">• Top-level business object• Request business object• Response business object (optional)• Fault business object (optional)	“Creating the payload business-object structure for v4.2.2 ICS over HTTP”
If you are using Backend Integration packaging for your message	Add to the payload business object the business objects to hold the message header information: <ul style="list-style-type: none">• Dynamic meta-object• HTTP-properties business object	“Creating HTTP transport-level header information for v4.2.2 ICS” on page 106.
If the document includes attachments	You must also create additional business objects to represent the attachments.	“Creating attachment-related business object definitions” on page 60

Creating the payload business-object structure for v4.2.2 ICS over HTTP

The Adapter for HTTP expects a payload business-object structure that consists of the following business objects:

- A top-level business object
- A request business object
- A fault business object (optional)
- A response business object (optional)

Figure 18 shows a sample business-object structure for a payload business object definition for use with a v4.2.2 InterChange Server over the HTTP transport protocol.

Note: For a detailed description of this business-object structure, refer to the *Adapter for HTTP User Guide*.

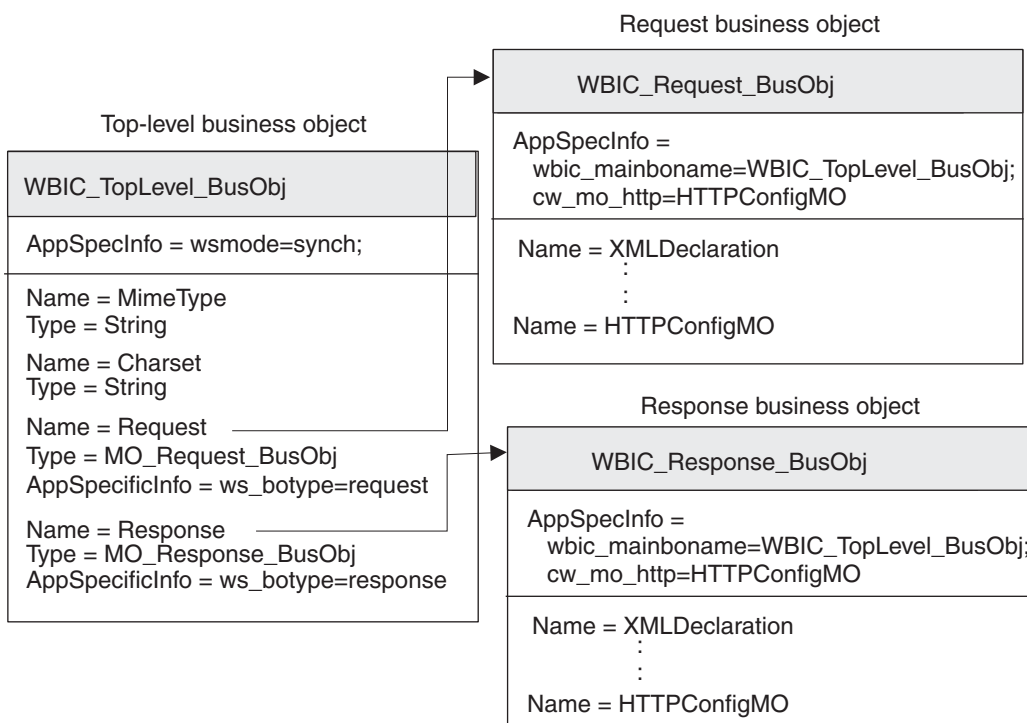


Figure 18. Business-object structure for the HTTP payload business object for v4.2.2 ICS

Top-level business object: The top-level business object is a wrapper for the request and response business objects. You must create a business object definition for this business object. Table 57 summarizes the attributes of the top-level business object definition.

Table 65. Attributes of top-level business object

Attribute	Attribute type	Description
MimeType	String	Defines the content type and format of the data that is being passed to the URL.
Charset	String	Used to determine which data handler to call.

Table 65. Attributes of top-level business object (continued)

Attribute	Attribute type	Description
Request	Business object	Child business object that represents the request message. The purpose of this business object depends on whether it participates in request processing or event notification. For more information on the structure of this business object, see “Request business object” on page 90.
Response	Business object	Child business object that represents the response message (if you are expecting a response). The purpose of this business object depends on whether it participates in request processing or event notification. For more information on the structure of this business object, see “Response business object” on page 105.

Note: When using the Adapter for HTTP with Business Integration Connect, do *not* need to include fault business objects in your top-level business object.

Table 66 summarizes the application-specific information that the top-level business object definition can have.

Table 66. Application-specific information for the top-level business object definition

Application-specific information	Tag	Description
Business-object level	ws_mode	Defines whether the interaction is synchronous or asynchronous
Attribute level	ws_botype	Defines which attribute contains the request or response business object

For a complete description of the structure of the top-level business object and its application-specific information, see the *Adapter for HTTP User Guide*.

Request business object: The request business object contains the data to be passed to the URL. It represents the HTTP request message. The purpose of this request business object depends on which InterChange Server task it is participating in, as follows:

- For event notification (sending a document to InterChange Server), the request business object contains the request message from Business Integration Connect, which is the event to be sent to InterChange Server.
- For request processing (receiving a document from InterChange Server), the request business object contains the request that InterChange Server is making to Business Integration Connect.

Note: The top-level business object identifies its child business objects as its request and response business objects. However, this structure is used in *both* request processing and event notification,

For the basic description of the request business object's structure, refer to the *Adapter for HTTP User Guide*. For use with Business Integration Connect, there are two customizations you must make to the structure of the request business object definition:

- If the document that Business Integration Connect sends to InterChange Server uses Backend Integration packaging, you must add to the request business object definition a special attribute to identify the HTTP protocol-configuration meta-object.

This attribute provides configuration information for the transport-level headers of the message. For more information, see "Creating HTTP transport-level header information for v4.2.2 ICS" on page 106.

- To the business-object-level application-specific information of the request business object definition, add the tags shown in Table 67.

Table 67. Tags in application-specific information of request business object

Application-specific-information tag	Description	Required?
ws_tloname	Gives the name of the top-level business object	Only required if business object definition participates in event notification
cw_mo_http	Specifies the HTTP protocol-configuration meta-object, which contains the HTTP transport-level header fields. For more information, see "Creating HTTP transport-level header information for v4.2.2 ICS" on page 106.	Only required if you are using Backend Integration packaging

Note: If you are using the Attachment data handler to process documents wrapped in an XML transport envelope, you must modify your request business object to hold the attachments, as described in "Creating attachment-related business object definitions" on page 60.

Response business object: The response business object contains the data to be received from the URL. It contains attributes for the various XML tags in the response message. The purpose of this response business object depends on which InterChange Server task it is participating in, as follows:

- For event notification, the response business object contains the response message, which is sent from the collaboration in InterChange Server.
- For request processing, the response business object contains the information from Business Integration Connect in response to the request that InterChange Server sent.

Regardless of whether the response is part of event notification or request processing, a response business object is sent *only* if the exchange between Business Integration Connect and InterChange Server is *synchronous* and a business response is expected in response to your request.

For the basic description of the fault business object's structure, refer to the *Adapter for HTTP User Guide*. For use with Business Integration Connect, there are customizations you must make to the structure of the request business object definition:

- If the document that Business Integration Connect sends to InterChange Server uses Backend Integration packaging, you must add to the response business object definition a special attribute to identify the HTTP protocol-configuration meta-object.
This attribute provides configuration information for the transport-level headers of the message. For more information, see “Creating HTTP transport-level header information for v4.2.2 ICS.”
- To the business-object-level application-specific information of the response business object definition, add the tags shown in Table 67 on page 105.
- In the top-level business object, add the `ws_botype` tag to the attribute-level application-specific information for the attribute that corresponds to the response business object.

This tag has the following syntax:

```
ws_botype=response
```

- Decide whether to add the `wbic_response_mime` business-object level application-specific information. This application-specific information is optional. It specifies the MIME type for the data handler to be used for the response business object.

If this tag is *not* specified, the Wrapper data handler uses the child meta-object indicated by the `wbic_response_mime` attribute (in the top-level business object) to determine the data handler to use for the response.

If the exchange between Business Integration Connect and InterChange Server is *asynchronous*, Business Integration Connect does *not* expect a response, so you do not need to create a response business object.

Creating HTTP transport-level header information for v4.2.2 ICS

If you are sending documents with Backend Integration packaging over the HTTP transport protocol, your request business object needs to contain custom transport-level header information. The Adapter for HTTP expects this custom header information to be in a **dynamic meta-object**.

Figure 19 shows the business-object structure for a request business object that represents a Business Integration Connect document with Backend Integration packaging over the HTTP transport protocol.

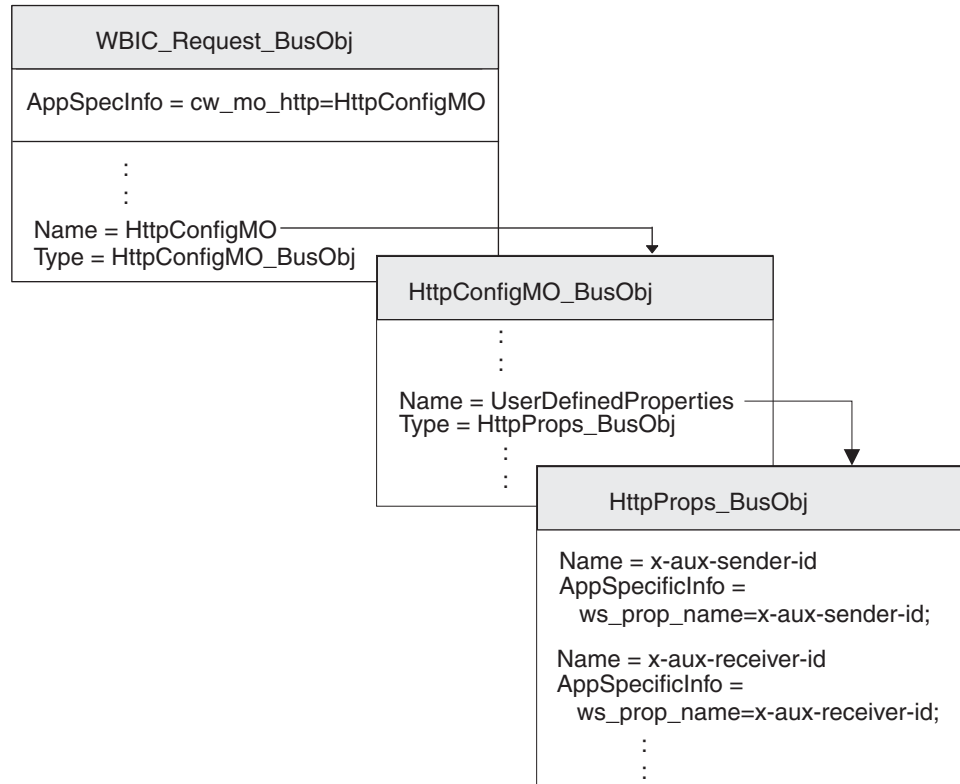


Figure 19. Relationship of the request business object to the HTTP protocol-configuration meta-object

Make sure your business-object structure includes an HTTP protocol-configuration meta-object by taking the following steps:

1. Create a business object definition to hold the HTTP properties required by the Backend Integration packaging.
2. Create a business object definition for the HTTP protocol-configuration meta-object.
3. Modify the business object definition for your request business object to include an attribute for the HTTP protocol-configuration meta-object.

Each of these steps is described in the sections below.

Creating the user-defined-properties business object: The Adapter for HTTP supports a **user-defined-properties business object** to hold custom properties in the HTTP protocol-configuration meta-object. Business Integration Connect uses this business object to hold HTTP properties required by the Backend Integration packaging. It can also contain the Content-Type attribute, which specifies the content-type header to set in the request message, and the content-length attribute, which specifies the length of the message, in bytes. Table 4 on page 11 describes each of the valid transport-header fields.

To create a user-defined-properties business object definition for the HTTP header fields, take the following steps:

1. Create an attribute within the business object definition for each of the transport-header fields.

All attributes should have an attribute type of String. You can name the attribute with the exact name of the HTTP property (as listed in the Header field column of Table 4 on page 11).

2. For each of the attributes in the HTTP-properties business object, add application-specific information to identify the purpose of the associated attribute.

This attribute-level application-specific information has the following format:

```
ws_prop_name=HTTPproperty
```

where *HTTPproperty* is one of the values in the Header field column of Table 4 on page 11.

In Figure 19 on page 107, the `HttpProps_BusObj` business object definition contains attributes for the various transport-header fields. These attributes all have attribute-level application-specific information to specify the name of the related protocol header. For example, the `x-aux-sender-id` attribute has the application-specific information set as follows:

```
ws_prop_name=x-aux-sender-id
```

Creating the HTTP protocol-configuration meta-object: For event notification, the request, response, or fault business object can contain a dynamic meta-object called the **HTTP protocol configuration meta-object** to hold configuration information (such as header information).

For the basic description of the HTTP protocol-configuration business object's structure, refer to the *Adapter for HTTP User Guide*. For use with Business Integration Connect, you must make the following customizations to the structure of the HTTP protocol-configuration business object definition:

1. Create an attribute within the business object definition for any of the fields you require.

All attributes should have an attribute type of String.

Note: For a complete list of attributes in the HTTP protocol-configuration meta-object, see the *Adapter for HTTP User Guide*.

2. Add the `UserDefinedProperties` attribute to this business object definition. The attribute type of this attribute is the business object definition for the user-defined-properties business object (see "Creating the user-defined-properties business object" on page 107).

For example, in Figure 19 on page 107, the `HttpConfigMO_BusObj` business object definition contains the `UserDefinedProperties` attribute, whose attribute type is `HttpProps_BusObj`.

Modify the request business object definition: The request business object definition represents the information requested from Business Integration Connect. For information on how to create the request business object, see "Request business object" on page 104. To incorporate the dynamic meta-object into your payload business-object structure, you must make the following modifications to your request business object definition:

1. Add an attribute to your request business object definition to hold the HTTP protocol-configuration meta-object.

The attribute type for this attribute is the business object definition for the HTTP protocol-configuration meta-object (see "Creating the HTTP protocol-configuration meta-object").

2. Add the `cw_mo_http` tag to the business-object-level application-specific information of your request business object definition to identify the attribute that contains the HTTP protocol-configuration meta-object.

The `cw_mo_http` tag has the following format:

```
cw_mo_http=HttpConfigMetaObjAttr
```

where `HttpConfigMetaObjAttr` is the name of the attribute in the request business object that holds the HTTP protocol-configuration meta-object.

For example, in Figure 19 on page 107, an attribute named `HttpConfigM0` has been added to the request business object definition, `WBIC_HttpRequest_BusObj`. This attribute contains the dynamic meta-object, which is a child business object of type `HttpConfigM0_BusObj`. In addition, the application-specific information of the request business object has been modified to include the following `cw_mo_http` tag to identify this dynamic meta-object:

```
cw_mo_http=HttpConfigM0
```

Creating v4.2.2 ICS artifacts for HTTP

To configure a v4.2.2 InterChange Server for communication with Business Integration Connect over the HTTP transport protocol, you must create the InterChange Server artifacts shown in Table 68.

Table 68. Artifacts for communicating with v4.2.2 ICS over the HTTP transport protocol

ICS artifact	Purpose	For more information
Business object definitions	Represent the document	“Creating business object definitions for v4.2.2 ICS over HTTP” on page 102
Connector object	Represents the Adapter for HTTP at run-time	“Creating the HTTP connector object”
Collaboration template and collaboration object	Represents the business process that InterChange Server uses to process the document	“Binding collaborations to communicate with Adapter for HTTP” on page 110

Creating the HTTP connector object

To obtain an instance of the Adapter for HTTP at run-time, you must take the following steps within System Manager:

1. Create the connector objects:
 - Create a connector object to represent an instance of the Adapter for HTTP.

Note: In the Supported Business Objects tab of Connector Configurator, make sure that you specify all business object definitions you created for use with the Adapter for HTTP. For a description of these business object definitions, see “Creating business object definitions for v4.2.2 ICS over HTTP” on page 102.

- If required by your collaboration, create a connector object for the Port Connector.
2. Configure the connector objects

For information on how to configure your Adapter for HTTP connector object for use with Business Integration Connect, see “Setting up the environment for HTTP transport with v4.2.2 ICS” on page 99.

Binding collaborations to communicate with Adapter for HTTP

As described in “Creating the collaborations” on page 48, a collaboration object must exist at run-time for InterChange Server to know where to receive and send business objects. When you create the collaboration object for the collaboration that uses the Adapter for HTTP to send information to and receive it from Business Integration Connect, you bind the collaboration ports, as follows:

- For request processing: the “to” port, which sends requests to Business Integration Connect, should be set to the connector object you created for the Adapter for HTTP; that is, the Adapter for HTTP is the *destination* adapter.
- For event notification: the “from” port, which receives events from Business Integration Connect, should be set to the connector object you created for the Adapter for HTTP; that is, the Adapter for HTTP is the *source* adapter.

Sending SOAP documents over HTTP/S

SOAP documents differ from other types of documents exchanged over HTTP/S. They use the standard Adapter for Web Services, which calls the SOAP data handler to transform SOAP messages into business objects and to transform business objects into SOAP messages. This section describes how to send and receive SOAP documents between WebSphere Business Integration Connect and WebSphere InterChange Server over the HTTP transport protocol

Notes:

1. To send and receive non-SOAP documents between WebSphere Business Integration Connect and a pre-4.2.2 WebSphere InterChange Server over the HTTP transport protocol, see “Using HTTP transport protocol with pre-4.2.2 ICS” on page 67.
2. To send and receive non-SOAP documents between WebSphere Business Integration Connect and a v4.2.2 WebSphere InterChange Server over the HTTP transport protocol, see “Using HTTP transport protocol with v4.2.2 ICS” on page 95.

Refer to the Adapter for Web Services documentation for information on the business-object structure and on the WSDL Object Discovery Agent (ODA), a design-time tool you can use to generate SOAP business objects that include information about the target web services.

As described in the Administrator Guide, you must have set up a target to receive Web service invocations from a back-end system (the Web services target) as well as a target to receive Web service invocations from a community participant (the external Web services target).

Components required for sending and receiving

To send a SOAP document from Business Integration Connect to InterChange Server using the HTTP transport protocol, you use the components listed in Table 69. All of these components are provided as part of the Business Integration Connect release.

Table 69. Components required to send SOAP documents to InterChange Server through HTTP

Component	Description	Notes and restrictions
WebSphere Business Integration Adapter for Web Services	This adapter allows InterChange Server to exchange business objects with applications that send or receive data in the form of HTTP messages.	<ol style="list-style-type: none"> 1. This adapter <i>cannot</i> be used with non-SOAP documents. 2. Make sure you are using the Adapter for Web Services 3.1.0 (or higher). Refer to the <i>Adapter for Web Services User Guide</i> to make sure that the level of the adapter is compatible with the version of WebSphere InterChange Server you are using.

Note: If a SOAP document contains attachments, you do not need to use the Attachment data handler to handle them.

How a community participant invokes a Web Service

The following steps occur when a community participant sends a request for a collaboration that is exposed as a Web Service that the Community Manager provides:

1. The community participant sends a SOAP request message to the destination specified in the WSDL document generated for the collaboration. Note that the endpoint specified in the WSDL is the Web services target (URL) of Business Integration Connect instead of the actual endpoint.
2. Business Integration Connect receives and routes the message to the Adapter for Web services.
3. The Adapter for Web Services sends the SOAP message to the SOAP data handler to convert the SOAP message to a business object. The adapter invokes the collaboration exposed as a web service.
4. If this is a request/response operation, the collaboration returns a SOAP Response (or Fault) business object.
5. If the collaboration returned a SOAP Response (or Fault) business object, the Adapter for Web Services calls the SOAP data handler to convert the SOAP Response (or Fault) business object to a SOAP response message. The adapter returns the response to Business Integration Connect. If the collaboration did not return a SOAP response (or Fault) business object, the Adapter for Web Services returns the appropriate HTTP response status code.
6. Business Integration Connect routes the response to the Web service.

How the Community Manager invokes a Web Service

The Public WSDL provided by Business Integration Connect can be used for creating business objects using WSDL ODA. It is important to note that when the Web service is provided by a community participant for use by the Community Manager, the public URL used by the Community Manager to invoke the Web service should contain the following query string:

`?to=<Community participant Web Service Provider's business ID>`

For example, the following address tells Business Integration Connect that the provider of the Web service is the participant with business ID 123456789:

`http://WBICHost/bcgreceiver/Receiver?to=123456789`

The WSDL ODA will not add the query string in the default value of the URL attribute of the Web Service top-level business object.

The following steps occur when a collaboration sends a request (to the Adapter for Web Services) to invoke a Web Service of a community participant:

1. The collaboration sends a service call request to the adapter, which calls the SOAP data handler to convert the business object to a SOAP request message.
2. The adapter invokes the web service by sending the SOAP message to the external Web services target (URL) on Business Integration Connect.
3. Business Integration Connect acts as a proxy, sending the SOAP message to the endpoint corresponding to the destination (community participant) Web service. This invokes the Web service.
4. The invoked web service receives the SOAP request message and performs the requested processing.
5. The invoked Web service sends a SOAP response (or fault) message. In the case of a one-way operation, the appropriate HTTP status code is returned.
6. If this is a request/response Web Service, Business Integration Connect routes the SOAP response (or fault) message to the adapter, which calls the data handler to convert it to a response or fault business object. The connector returns the SOAP response or fault business object to the collaboration.

Chapter 4. Integrating with InterChange Server over JMS

This chapter describes how to integrate WebSphere Business Integration Connect with WebSphere InterChange Server over the JMS transport protocol. It provides information on how to configure InterChange Server and the ICS-compatible components required for communication over JMS.

Note: For information on how to configure WebSphere Business Integration Connect to communicate with InterChange Server over JMS, see “Configuring Business Integration Connect for InterChange Server” on page 41. For general information on how to configure InterChange Server, see “Configuring InterChange Server” on page 44.

This chapter provides the following information on how to send and receive documents between WebSphere Business Integration Connect and WebSphere InterChange Server through the use of the JMS transport protocol:

- “Components required for documents over JMS transport”
- “Setting up the environment for JMS transport” on page 119
- “Creating business object definitions for JMS” on page 121

Components required for documents over JMS transport

For Business Integration Connect to communicate with InterChange Server over the JMS transport protocol requires that these two components be configured to work with JMS. Table 70 summarizes these configuration steps.

Table 70. Configuring Business Integration Connect and InterChange Server for JMS transport protocol

Component	Version	For more information
WebSphere Business Integration Connect	4.2.2	“Configuring for outgoing documents over JMS transport protocol” on page 42 “Configuring for incoming documents over JMS transport protocol” on page 43
WebSphere InterChange Server	4.1.1, 4.2.0, 4.2.1, 4.2.2	“Creating ICS artifacts for JMS” on page 125

In addition, to send or receive a document between Business Integration Connect and InterChange Server over the JMS transport protocol, you also use the ICS-compatible components listed in Table 71..

Table 71. Components required to transfer documents to and from InterChange Server through JMS

Component	Description	Notes and restrictions
WebSphere Business Integration Adapter for JMS (Adapter for JMS)	This adapter allows InterChange Server to exchange business objects with applications that send or receive data in the form of JMS messages. The Adapter for JMS and Business Integration Connect communicate through JMS queues.	<ol style="list-style-type: none"> 1. Make sure you are using the Adapter for JMS Version 2.3.1 (or higher), which provides support for custom header properties. Refer to the adapter documentation to make sure that the version of the adapter is compatible with the version of InterChange Server you are using. 2. The Adapter for JMS supports JMS Text messages only. If you intend to use JMS Byte messages, use the Adapter for JMS Version 2.5.0.
A payload data handler	This data handler converts the payload between its document format and its business-object representation.	For more information, see “Specifying the payload data handler” on page 120.
Attachment data handler	This data handler converts documents with attachments into business objects.	This data handler is required <i>only</i> if your documents include attachments. For more information, see “Handling documents with attachments” on page 49.

The following sections describe how the components in Table 71 work together to send and receive documents between Business Integration Connect and InterChange Server over the JMS transport protocol.

Sending documents over JMS transport

For Business Integration Connect to send a document to InterChange Server using the JMS transport protocol, you use the Adapter for JMS to retrieve the message that Business Integration Connect has put on a JMS queue. The adapter then routes the message to InterChange Server. Figure 20 provides an overview of how Business Integration Connect sends documents to InterChange Server over the JMS transport protocol.

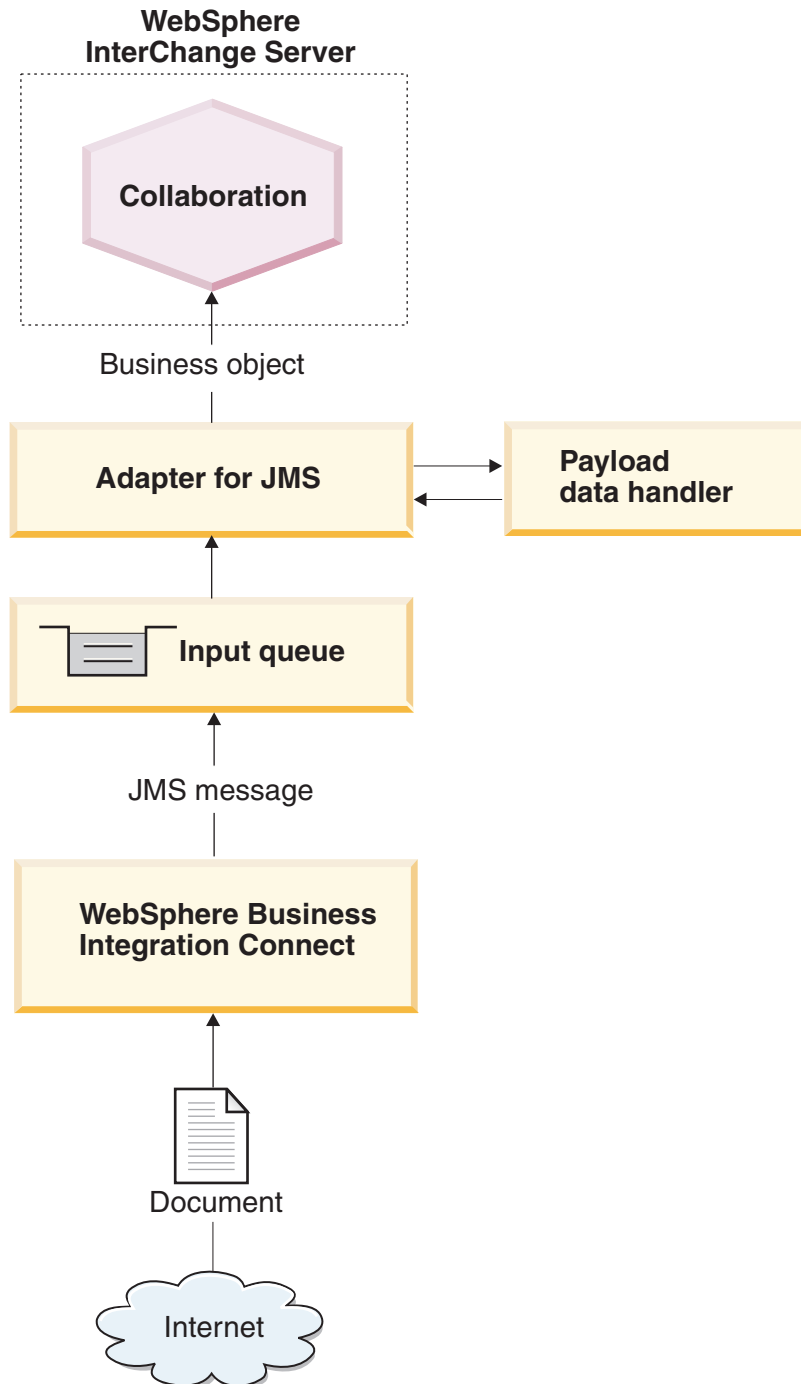


Figure 20. Message flow from Business Integration Connect to a collaboration through the JMS transport protocol

The following steps describe how Business Integration Connect participates in event notification by sending a document to a collaboration within InterChange Server over the JMS transport protocol:

1. Business Integration Connect posts a message to its JMS outbound queue.
If the packaging type of the document was Backend Integration, Business Integration Connect has provided custom properties in this message. The JMS message header, JMSType, is set with the content type of the payload.

Note: Within Business Integration Connect, you must configure a gateway that identifies the JMS queue to which Business Integration Connect sends the message and on which the Adapter for JMS is polling. For more information, see “Configuring for outgoing documents over JMS transport protocol” on page 42.

2. When the Adapter for JMS sees a message on one of its input queues, it retrieves the message.

The JMS queue that Business Integration Connect uses as its outbound queue is the same queue that the Adapter for JMS uses as its input queue. For information on how to set up this queue, see “Configuring the JMS queues” on page 119. For detailed information on the processing of the Adapter for JMS, see the *Adapter for JMS User Guide*.

3. The Adapter for JMS moves the message to its in-progress queue.
4. The Adapter for JMS extracts the body of the JMS message and invokes a data handler with the body of the message. This data handler converts the body of the JMS message to a business object.

Note: If your messages have attachments, you can install the Attachment data handler and then configure the Adapter for JMS to call it to convert the body of the JMS message to a business object. For more information, see “Handling documents with attachments” on page 49.

When Backend Integration is the packaging type and the document contains attachments, the configured data handler is responsible for handling the payload and attachments.

5. The data handler returns the business object to the Adapter for JMS.

Note: If the Attachment data handler was used, this business object contains the payload as well as the attachments.

6. If the Adapter for JMS finds a child dynamic meta-object (specified using `cw_mo_conn` in the business-object level application specific information), the adapter populates the user-defined JMS headers present in the business object with the headers present in the JMS message.
7. The Adapter for JMS delivers the business object to the InterChange Server as part of a subscription delivery.

Receiving documents over JMS transport

For Business Integration Connect to receive a document from InterChange Server using the JMS transport protocol, you use the Adapter for JMS, which places the message it receives from InterChange Server on a JMS queue for Business Integration Connect to retrieve. Figure 21 provides an overview of how Business Integration Connect receives documents from InterChange Server over the JMS transport protocol.

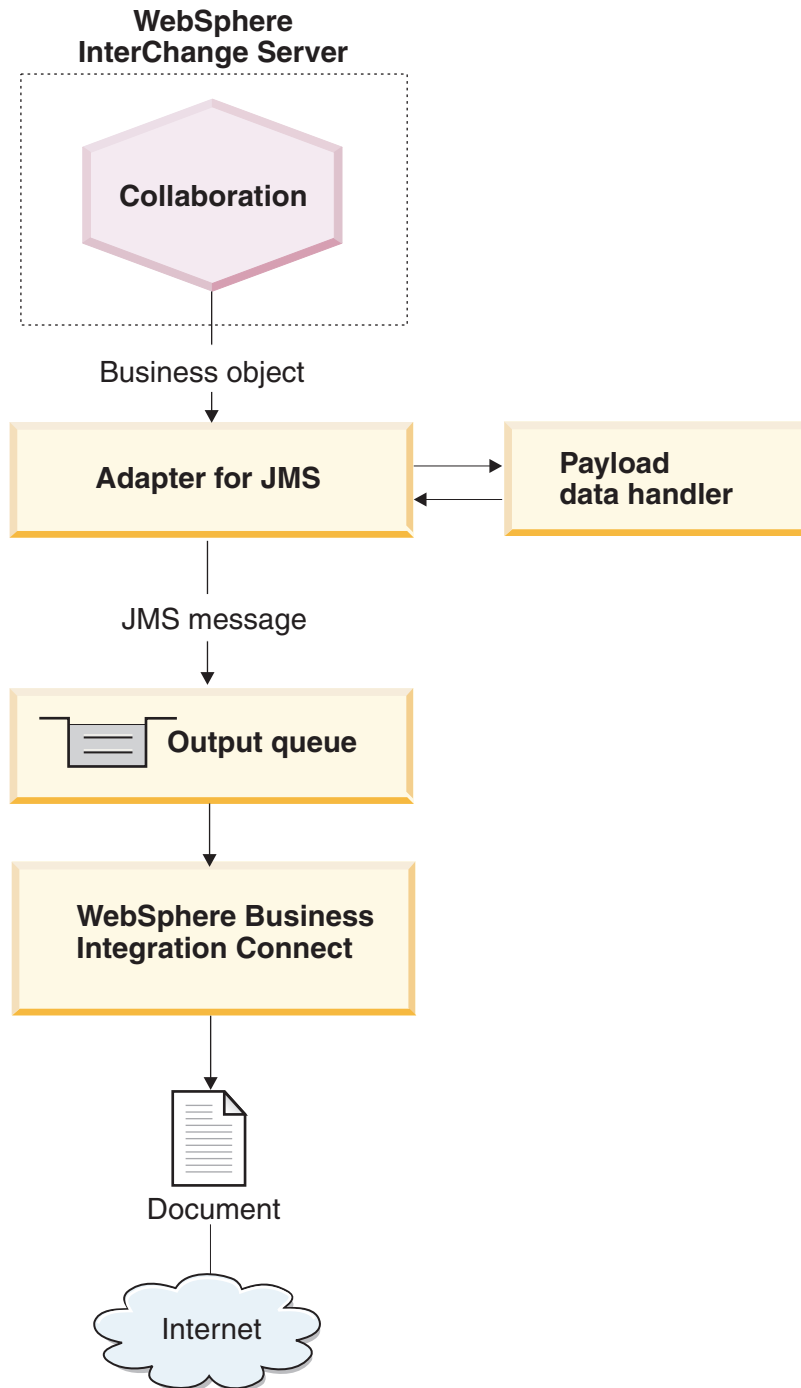


Figure 21. Message flow from a collaboration to Business Integration Connect through the JMS transport protocol

The following steps describe how Business Integration Connect participates in request processing by receiving a document from a collaboration within InterChange Server over the JMS transport protocol:

1. The collaboration within InterChange Server makes a service call to the Adapter for JMS, sending it the request business object.

The request business object contains application-specific information pointing to a dynamic meta-object that contains the JMS transport-level header information, which Business Integration Connect expects.

2. The Adapter for JMS uses a data handler to convert the business object that the collaboration has sent it into a JMS message.

The adapter reads the `DataHandlerMimeType` and `DataHandlerConfigMO` properties to determine the data handler to use. For more information, see “Specifying the payload data handler” on page 120.

Note: If your documents have attachments, install the Attachment data handler and then configure the Adapter for JMS to call it to convert the request business object to a document with attachments. For more information, see “Handling documents with attachments” on page 49.

3. The data handler converts the business object to a string and returns it to the Adapter for JMS.
4. The Adapter for JMS determines, from the request business object, the name of the dynamic meta-object for custom JMS properties.

The adapter searches the application-specific information of the request business object for the `cw_mo_conn` tag, which identifies the attribute that contains the dynamic meta-object. If you are using Backend Integration packaging for your document, you can specify transport-level header information in this dynamic meta-object.

5. The Adapter for JMS searches the dynamic meta-object for the `JMSProperties` attribute.

If this attribute is populated, the adapter sets the transport-level header fields in the request document. Within the `JMSProperties` attribute, you can also specify the content-type standard JMS header. For more information, see “Creating JMS header information” on page 122.

6. The Adapter for JMS creates a JMS message, using the string returned by the data handler. It also sets any custom properties, as defined in the dynamic meta-object.

Note: Versions of the Adapter for JMS *before* 2.4.1 can write only JMS text messages.

7. The Adapter for JMS sends the resulting request message to an output queue. The queue can be specified in the static meta-object or the dynamic meta-object. For information on specifying queues, see “Identifying the JMS queues” on page 121. Business Integration Connect listens on this JMS queue, which is configured as its inbound queue in its target definition. For more information, see “Configuring for incoming documents over JMS transport protocol” on page 43.
8. Business Integration Connect receives the message from its JMS inbound queue, as configured in its target.

Note: Business Integration Connect supports only *asynchronous* interaction with InterChange Server over JMS. Therefore, you might not want to wait for the response. The response from the community participant or Business Integration Connect can come on a different queue. You can configure the Adapter for JMS to poll that queue. The response that comes on the queue can be delivered to InterChange Server as part of the event delivery.

Setting up the environment for JMS transport

Because the sending and receiving of documents to and from InterChange Server involves ICS-compatible components, you must perform the setup and configuration tasks described in Table 72. For information on how to configure Business Integration Connect for use with InterChange Server over JMS, see “Configuring Business Integration Connect for InterChange Server” on page 41.

Table 72. Setting up the environment for use of JMS transport protocol

Configuration step	For more information
1. Configure your JMS queues.	“Configuring the JMS queues”
2. Configure the WebSphere Business Integration Adapter for JMS.	“Configuring the Adapter for JMS” on page 120

Note: If your documents contain attachments, you must also install and configure the Attachment data handler. For more information, see “Handling documents with attachments” on page 49.

Configuring the JMS queues

To use the JMS transport protocol with InterChange Server, you must set up the JMS system that WebSphere MQ provides. Supported versions of InterChange Server use version 5.3 of WebSphere MQ as a JMS provider. Therefore, you can use the steps in “Configuring a JMS protocol with WebSphere MQ,” on page 173 to set up the JMS transport-protocol mechanism.

Important: The steps in “Configuring a JMS protocol with WebSphere MQ,” on page 173 must be performed on the machine on which WebSphere Business Integration Connect resides. This guide assumes that the JMS transport-mechanism required by the Adapter for JMS and InterChange Server has already been set up as part of the InterChange Server installation.

When you create your JMS queues for use between Business Integration Connect and InterChange Server, consider the following points:

- Part of the InterChange Server installation process involves the creation of a WebSphere MQ queue manager. You can use this queue manager with Business Integration Connect.
- When you create your JMS queue aliases, you might want to name them to indicate the direction of flow between Business Integration Connect and InterChange Server. For example, if you create the queues listed in the Original queue name column of Table 73, you could rename these queues to indicate the InterChange Server directionality, as shown in the Directional queue name column of Table 73.

Table 73. Naming JMS queues for InterChange Server directionality

Original queue name	Directional queue name
inQ	ICS2WBIC
outQ	WBIC2ICS

Configuring the Adapter for JMS

The Adapter for JMS is the ICS-compatible component that allows Business Integration Connect to exchange documents with InterChange Server in the form of JMS messages. It supports the following interactions with InterChange Server:

- For request processing, it receives the request business object from InterChange Server, converts it to a JMS message (using a data handler), and puts the JMS message on a JMS queue (see Figure 21 on page 117), where it can be picked up by Business Integration Connect.
- For event notification, it polls a JMS queue for JMS messages from Business Integration Connect. When it finds a JMS message, it converts it to an event business object (using a data handler) and sends it to InterChange Server.

Important: WebSphere Business Integration Connect does *not* include the WebSphere Business Integration Adapter for JMS. You must obtain this product separately and install it according to the instructions in its *Adapter for JMS User Guide*. It is important that you read the steps described in this guide to correctly install and configure your Adapter for JMS.

When you have configured the Adapter for JMS to communicate with InterChange Server, follow the steps in this section to configure this adapter to accept JMS messages from Business Integration Connect:

- “Specifying the payload data handler”
- “Identifying the JMS queues” on page 121

Specifying the payload data handler

As Figure 21 on page 117 shows, the Adapter for JMS uses a data handler to convert the business objects it receives from InterChange Server into the appropriate JMS messages.

Note: The data handler that the Adapter for JMS calls converts the payload of the document. If your document is wrapped in an XML transport envelope (it contains attachments or the Envelope Flag is Yes), configure the Attachment data handler as the payload data handler. For more information, see “Handling documents with attachments” on page 49.

To indicate which data handler to use to convert the payload, you must take the steps listed in “Business object conversion” on page 46. In addition, you must configure the Adapter for JMS to use this payload data handler. In Connector Configurator, take the following steps:

- Set the following connector configuration properties to identify the payload data handler:
 - Set the `DataHandlerConfigMO` and `DataHandlerMimeType` properties with the name of the top-level data-handler meta-object and the supported MIME type, respectively.
 - Set the `DataHandlerClassName` property with the name of the data-handler class to instantiate.

Note: You set *either* the `DataHandlerConfigMO` and `DataHandlerMimeType` properties *or* the `DataHandlerClassName` property.

- Include the top-level data-handler meta-object in the list of supported business objects.

You can also specify the data handler to use in the static or dynamic meta-object. The same properties (`DataHandlerMimeType`, `DataHandlerConfigMO`, and `DataHandlerClassName`) are available as attributes in these meta-objects. For a complete description, refer to the *Adapter for JMS User Guide*.

Identifying the JMS queues

When the Adapter for JMS receives a document from InterChange Server, puts the message in its outbound queue, which is the one that Business Integration Connect's Receiver is polling. Similarly, when Business Integration Connect sends a document to InterChange Server, it puts the document in its outbound queue, which is the one that the Adapter for JMS is polling.

Table 74 summarizes how to configure the JMS queues that the Adapter for JMS uses to receive and send documents.

Note: For a complete description of how to configure JMS queues, refer to the *Adapter for JMS User Guide*.

Table 74. JMS queues

JMS queue	Configuration set
Input queue	<p>Set the <code>InputDestination</code> connector configuration property to the name of the JMS queue that the Adapter for JMS will poll for incoming messages.</p> <p>Make sure that the name of this queue is the same as the one Business Integration Connect is using as its JMS outbound queue. If this queue is not specified in <code>InputDestination</code>, the Adapter for JMS will <i>not</i> poll the queue.</p> <p>Note: The <code>InputDestination</code> property contains a comma-separated list of input queues. If the Adapter for JMS polls multiple queues, make sure that this list includes the name of the JMS queue that Business Integration Connect is using as its JMS outbound queue.</p>
Output queue	<p>At run-time, the collaboration can dynamically set the <code>OutputQueue</code> attribute in the dynamic meta-object to the name of the JMS queue that the Adapter for JMS will send its outgoing message.</p>

You must make sure that the static or dynamic meta-objects are configured so that they can write to the queue on which the Business Integration Connect target is listening.

Creating business object definitions for JMS

The Adapter for JMS sends and receives your document to InterChange Server in the form of a payload business object. The Adapter for JMS invokes the payload data handler to handle this business object when it receives or sends a Business Integration Connect document, as follows:

- For request processing, the payload data handler converts the request business object to its corresponding JMS message.
- For event notification, the data handler converts the JMS message to an event business object.

Therefore, you must create the business object definitions shown in Table 75. to represent the payload business-object structure that the Adapter for JMS expects.

Table 75. Business object definitions for the Adapter for JMS

Condition	Business object definition	For more information
If you are using None or Backend Integration packaging for your message <i>and</i> your documents do <i>not</i> have attachments	Payload business object	“Creating the payload business-object structure for JMS.”
If you are using Backend Integration packaging for your document	Business objects to hold the message header information: <ul style="list-style-type: none"> • Dynamic meta-object • JMS-properties business object 	“Creating JMS header information.”
If the document includes attachments	You must also create additional business objects to represent the attachments.	“Creating attachment-related business object definitions” on page 60

Creating the payload business-object structure for JMS

The structure of the payload business object for the JMS transport protocol depends on the packaging type, as follows:

- If your document uses None packaging, there are no special requirements to create the payload business object for a document sent over the JMS transport protocol.
As discussed in “Business object for the document” on page 45, you must create an attribute for each piece of payload information you need to transfer.
- If your document uses Backend Integration packaging, you must take the following steps:
 - Add to the payload business object definition a special attribute to identify the dynamic meta-object. This attribute provides configuration information for the transport-level headers of the message.
 - In the business-object-level application-specific information, add the `cw_mo_conn` tag to identify the attribute that contains the dynamic meta-object.

For more information on these steps, see “Creating JMS header information.”

Note: For request processing, the JMS transport protocol can *only* support asynchronous interactions. You can send a request business object, but you *cannot* obtain a response. Therefore, you must create a request business object definition but not a business object definition for a response.

Creating JMS header information

If you are sending or receiving documents that use Backend Integration packaging over the JMS transport protocol, your request business object needs to contain custom transport-level header information. The Adapter for JMS expects this custom header information to be in its **dynamic meta-object**.

Figure 22 shows the business-object structure that the Adapter for JMS uses for a request business object representing a Business Integration Connect document that uses Backend Integration packaging.

Note: The *Adapter for JMS User Guide* provides information about this required business-object structure. Refer to this guide when defining your business object definitions.

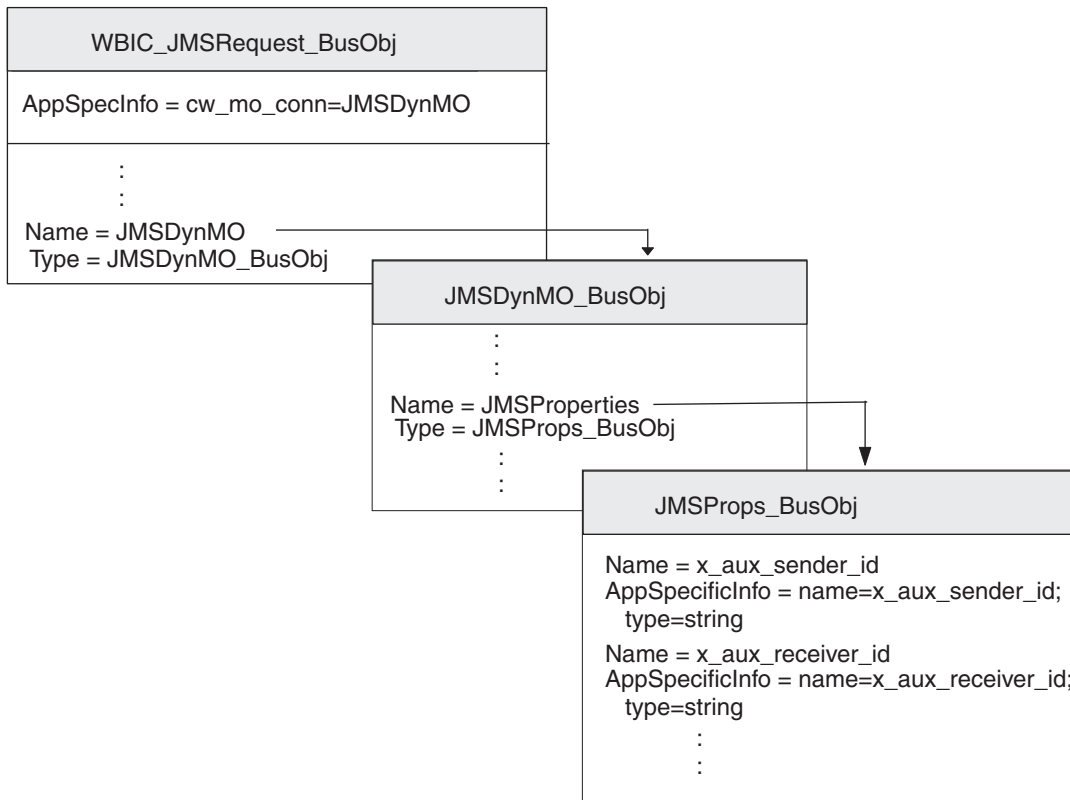


Figure 22. Relationship of the request business object to the JMS dynamic meta-object

Make sure your business-object structure includes a dynamic child meta-object by taking the following steps:

1. Create a business object definition to hold the JMS properties required by the Backend Integration packaging.
2. Create a business object definition for the dynamic meta-object.
3. Modify the business object definition for your request business object to include an attribute for the dynamic meta-object.

Each of these steps is described in the sections below.

Creating the JMS-properties business object

A **JMS-properties business object** contains JMS properties required for transport-level headings, which are needed by Backend Integration packaging. It can also contain the content-type attribute, which specifies the content-type header to set in the request message, and the content-length attribute, which specifies the length of the message, in bytes. Table 4 on page 11 describes each of the valid transport-header fields.

To create a JMS-properties business object definition, take the following steps:

1. Create an attribute within the business object definition for each of the transport-level header fields.

All attributes should have an attribute type of String. For JMS messages, the names of transport-header fields use underscores instead of hyphens, as shown in Table 76.

- For each of the attributes in the JMS-properties business object definition, add application-specific information to identify the purpose of the associated attribute.

This attribute-level application-specific information must have the following format:

```
name=JMSproperty;type=string
```

where *JMSproperty* is one of the values in the JMS property name column of Table 76.

- For any of the attributes in the JMS-properties business object definition, you can add a default value to indicate the common (or only valid) value for that transport-level field.

Table 76. Attributes for the JMS-properties business object definition

Transport-header field	JMS property name
x-aux-sender-id	x_aux_sender_id
x-aux-receiver-id	x_aux_receiver_id
x-aux-protocol	x_aux_protocol
x-aux-protocol-version	x_aux_protocol_version
x-aux-process-type	x_aux_process_type
x-aux-process-version	x_aux_process_version
x-aux-create-datetime	x_aux_create_datetime
x-aux-msg-id	x_aux_msg_id
x-aux-production	x_aux_production
x-aux-system-msg-id	x_aux_system_msg_id
x-aux-payload-root-tag	x_aux_payload_root_tag
x-aux-process-instance-id	x_aux_process_instance_id
x-aux-event-status-code	x_aux_event_status_code
x-aux-third-party-bus-id	x_aux_third_party_bus_id
x-aux-transport-retry-count	x_aux_transport_retry_count
content-type	content_type
content-length	content_length

Note: Table 76 does *not* provide an exhaustive list of the headers required for back-end integration. For a complete list and description of the headers, see “Transport-level header content” on page 11. Make sure you substitute underscore characters for any hyphens in transport-header field names.

In Figure 22 on page 123, the JMSProps_BusObj business object definition contains attributes for the various transport-level header fields. These attributes all have attribute-level application-specific information to specify the name of the related protocol header. For example, the `x_aux_sender_id` attribute has the application-specific information set as follows:

```
name=x_aux_sender_id;type=string
```

Creating the JMS dynamic meta-object

This **dynamic meta-object** contains a child business object with configuration information for the Adapter for JMS. To create a business object definition for a dynamic meta-object, take the following steps:

1. Create an attribute named `JMSProperties`, whose attribute type is the business object definition for the JMS-properties business object (see “Creating the JMS-properties business object” on page 123).
2. Add other configuration properties as appropriate. For a list of valid attributes in the dynamic meta-object, see the *Adapter for JMS User Guide*. Please consult this guide for information on how to create attributes to configure the dynamic meta-object.

For the Adapter for JMS to work with Business Integration Connect, the business object definition for the dynamic meta-object *must* include the attribute named `JMSProperties`, whose attribute type is the business object definition for the JMS-properties business object (see “Creating the JMS-properties business object” on page 123). For example, in Figure 22 on page 123, the `JMSDynMO_BusObj` business object definition contains attributes for various configuration properties (not shown) and includes the `JMSProperties` attribute.

Modifying the request business object definition

To incorporate the business-object structure into your request business object, you must make the following modifications to your request business object definition:

1. Add an attribute to your request business object definition to hold the dynamic child meta-object.

The attribute type for this attribute is the business object definition for the dynamic meta-object (see “Creating the JMS dynamic meta-object” on page 124).

2. Add the `cw_mo_conn` tag to the business-object-level application-specific information of your request business object definition to identify the attribute that contains the dynamic meta-object.

The `cw_mo_conn` tag has the following format:

```
cw_mo_conn=dynamicMetaObjAttr
```

where *dynamicMetaObjAttr* is the name of the attribute in the request business object that holds the dynamic meta-object.

For example, in Figure 22 on page 123, an attribute named `JMSDynMO` has been added to the request business object definition, `WBIC_JMSRequest_BusObj`. This attribute contains the dynamic meta-object, which is a child business object of type `JMSDynMO_BusObj`. In addition, the application-specific information of the request business object has been modified to include the following `cw_mo_conn` tag to identify this dynamic meta-object:

```
cw_mo_conn=JMSDynMO
```

Creating ICS artifacts for JMS

To configure InterChange Server for communication with Business Integration Connect over the JMS transport protocol, you must create the InterChange Server artifacts shown in Table 77.

Table 77. ICS artifacts for communicating over the JMS transport protocol

ICS artifact	Purpose	For more information
Business object definitions	Represent the document	“Creating business object definitions for JMS” on page 121
Connector object	Represents the Adapter for JMS at run-time	“Creating the JMS connector object” on page 126

Table 77. ICS artifacts for communicating over the JMS transport protocol (continued)

ICS artifact	Purpose	For more information
Collaboration template and collaboration object	Represents the business process that InterChange Server uses to process the document	“Binding collaborations to communicate with Adapter for JMS”

Creating the JMS connector object

To obtain an instance of the Adapter for JMS at run-time, take the following steps within System Manager:

1. Create the connector objects:
 - Create a connector object to represent an instance of the Adapter for JMS.

Note: In the Supported Business Objects tab of Connector Configurator, make sure that you specify all business object definitions you created for use with the Adapter for JMS. For a description of these business object definitions, see “Creating business object definitions for JMS” on page 121.
 - If required by your collaboration, create a connector object for the Port Connector.
2. Configure the connector objects.

For information on how to configure your Adapter for JMS for use with Business Integration Connect, see “Configuring the Adapter for JMS” on page 120.

Binding collaborations to communicate with Adapter for JMS

As described in “Creating the collaborations” on page 48, a collaboration object must exist at run-time for InterChange Server to know where to receive and send business objects. When you create the collaboration object for the collaboration that uses the Adapter for JMS to send information to and receive it from Business Integration Connect, you bind the collaboration ports, as follows:

- For request processing: the “to” port, which sends requests to Business Integration Connect, should be set to the connector object you created for the Adapter for JMS; that is, the Adapter for JMS is the *destination* adapter.
- For event notification: the “from” port, which receives events from Business Integration Connect, should be set to the connector object you created for the Adapter for JMS; that is, the Adapter for JMS is the *source* adapter.

Part 3. Integrating with other back-end systems

Chapter 5. Integrating with WebSphere Business Integration Message Broker

This chapter describes how to integrate WebSphere Business Integration Connect with WebSphere Business Integration Message Broker.

Notes:

1. For a description of the general process used to integrate Business Integration Connect with a back-end system, see Chapter 1, “Planning for back-end integration,” on page 3.
2. This chapter assumes that you are familiar with WebSphere Business Integration Message Broker and associated components, such as projects and message flows.

Often integration of WebSphere Business Integration Connect with a back-end system is done by two separate people or roles. Each role configures a particular component, for which that role has expertise. Therefore, this chapter separates the integration with WebSphere Business Integration Message Broker into the configuration of WebSphere Business Integration Connect and the configuration of Message Broker. Table 78 lists these configuration roles along with the places in this chapter to obtain the associated configuration information.

Table 78. Roles for Message Broker integration

Configuration role	For more information
Configuration of WebSphere Business Integration Connect	<ol style="list-style-type: none">1. “Planning for integration with Message Broker” on page 1302. “Configuring Business Integration Connect for Message Broker” on page 131
Configuration of WebSphere Business Integration Message Broker	<ol style="list-style-type: none">1. “Planning for integration with Message Broker” on page 1302. “Configuring Message Broker” on page 134

Note: While each of these configuration roles can be performed separately, each also requires common information so that the two components can communicate.

This chapter provides the following information:

- “Planning for integration with Message Broker” on page 130
- “Configuring Business Integration Connect for Message Broker” on page 131
- “Configuring Message Broker” on page 134
- “Using HTTP transport protocol with Message Broker” on page 135
- “Sending SOAP documents” on page 138
- “Using JMS transport protocol with Message Broker” on page 139

Planning for integration with Message Broker

To plan for your integration to WebSphere Business Integration Message Broker, follow the steps outlined in “Planning the back-end integration” on page 5. Table 79 summarizes the integration steps to integrate Business Integration Connect with Message Broker.

Table 79. Planning integration with WebSphere Business Integration Message Broker

Integration step	For more information
1. Confirm that you have a supported version of WebSphere Business Integration Message Broker installed and available to WebSphere Business Integration Connect.	Chapter 5: “Message Broker versions that Business Integration Connect supports”
2. Determine the business protocol of the WebSphere Business Integration Message Broker document.	Chapter 1: “What business protocol are you using?” on page 5
3. Determine the packaging type for the document: None or Backend Integration.	Chapter 1: “Which packaging will you use?” on page 10
4. Determine the message transport to use between WebSphere Business Integration Connect and WebSphere Business Integration Message Broker.	Chapter 5: “Message transports that Message Broker supports”
5. Configure WebSphere Business Integration Connect.	Chapter 5: “Configuring Business Integration Connect for Message Broker” on page 131

Message Broker versions that Business Integration Connect supports

Version 4.2.2 of Business Integration Connect can support integration with version 5.0 of Message Broker. Message Broker is available on several platforms, including Windows 2000 and several UNIX-based platforms. For more information, consult your installation guide for Message Broker in the WebSphere Business Integration Message Broker documentation set.

Message transports that Message Broker supports

Business Integration Connect supports the message-transport protocols shown in Table 10 on page 18. Of these supported protocols, the following two message-transport protocols are supported by WebSphere Business Integration Message Broker:

- HTTP transport protocol (including Web Services)
- JMS transport protocol

Support for these message transport protocols requires the installation and configuration of IBM WebSphere MQ.

Using HTTP with Message Broker

Message Broker uses the HTTP transport protocol for its Web Services transactions. To send and receive documents between Business Integration Connect and Message Broker over the HTTP protocol requires no additional software. However, to send the document out of Message Broker to some other destination does require WebSphere MQ.

Note: Business Integration Connect supports both asynchronous and synchronous interactions with Message Broker over HTTP.

Using JMS with Message Broker

Message Broker uses the JMS transport protocol for most of its transactions. To send and receive documents between Business Integration Connect and Message Broker over the JMS transport protocol, you must use JMS queues. If these two components reside on different machines, you must create the JMS queues on each machine. Basically, support for JMS involves the use of a message flow within Message Broker and the underlying JMS queues. For more information on how to configure for JMS, see “Using JMS transport protocol with Message Broker” on page 139.

Note: Business Integration Connect supports only asynchronous interactions with Message Broker over JMS.

Support for Message Broker integration

Business Integration Connect provides samples to assist you in the integration process with Message Broker. These samples reside in the following subdirectory of the Business Integration Connect product directory:

Integration/WBI/WBIMB/samples

Configuring Business Integration Connect for Message Broker

A general overview of how to configure Business Integration Connect to communicate with a back-end system is provided in “Configuring Business Integration Connect” on page 25. This section summarizes the steps needed to configure Business Integration Connect to communicate with Message Broker. To perform this configuration, use an instance of Business Integration Connect Enterprise or Advanced Edition that functions as the Community Manager in your hub community.

Configuration of Business Integration Connect involves the following steps:

- Configuring for support of outgoing documents
For information on sending documents from Business Integration Connect to Message Broker, see “Providing support for outgoing documents.”
- Configuring for incoming documents
For information on sending documents from Message Broker to Business Integration Connect, see “Providing support for incoming documents” on page 133.

Providing support for outgoing documents

For Business Integration Connect to send documents to any back-end system, you must perform the steps described in “Defining where to send the participant document” on page 27. When your back-end system is Message Broker, you need to create a gateway whose transport type matches the transport protocol used for messages between Business Integration Connect and Message Broker. When the Community Manager sends a document to Message Broker, it must know where to route the document. This location must conform with the transport protocol being used. The transport protocol must be one that Message Broker supports (see “Message transports that Message Broker supports” on page 130).

The following sections summarize how to create gateways for following transport protocols, which Message Broker supports:

- “Configuring for outgoing documents over HTTP transport protocol” on page 132

- “Configuring for outgoing documents over JMS transport protocol”

Configuring for outgoing documents over HTTP transport protocol

When the Community Manager sends a document to Message Broker over the HTTP protocol, the Community Manager routes the message through the defined gateway. This gateway identifies the URL where the document can be received by Message Broker. When Message Broker uses the HTTP protocol, it routes the document to the HTTPInput node of the message flow associated with the specified URL.

For the Community Manager to be able to send documents through a gateway over the HTTP transport protocol, you must create a gateway from the Gateway Details screen of the Community Console. This gateway must be configured to use the HTTP 1.1. transport protocol and to write to the URL on which the appropriate HTTPInput node is listening. As Table 80 shows, you provide this URL in the Target URI field of the gateway definition.

Note: An overview of how to create a gateway is provided in “Defining where to send the participant document” on page 27.

Table 80. HTTP values for Gateway Details screen for communication with Message Broker

Target Details field	Value	Notes and restrictions
Target URI	The URL should be the same as the one configured for the HTTPInput node in the Message Broker message flow	Obtain this URL from the configuration of the message flow in the WebSphere Business Integration Message Broker integration.

Configuring for outgoing documents over JMS transport protocol

When the Community Manager sends documents to Message Broker over the JMS protocol, the Community Manager routes the document to the appropriate JMS queue, where it can be transferred to the JMS queue from which Message Broker can retrieve it. For the Community Manager to obtain this JMS location, you must create a gateway in Business Integration Connect, one that uses the JMS transport protocol. This gateway must be configured to write to the queue whose contents are transferred to the queue on which Message Broker receives messages.

Note: For an overview of how to create a gateway, see “Defining where to send the participant document” on page 27.

For the Community Manager to be able to send documents through a gateway over the JMS transport protocol, create a gateway from the Gateway Details screen of the Community Console. When using WebSphere MQ version 5.3 as your JMS provider, use the information in Table 116 on page 179 to set the gateway fields. In addition, specify the information specified in Table 81 for the JMS protocol in the Gateway Details screen.

Table 81. JMS values for the Gateway Details screen for communication with Message Broker

Gateway Details field	Value	Notes and restrictions
JMS Queue Name	Name of the JMS queue, on the machine where Business Integration Connect resides	Documents received on this queue are transferred to the JMS queue on the machine where Message Broker resides.

Providing support for incoming documents

For Business Integration Connect to receive messages from any back-end system, you must perform the steps described in “Defining where to retrieve the back-end document” on page 31. When your back-end system is Message Broker, you need to take the following steps in your Community Manager:

1. As part of your participant profile for the Community Manager, define the gateway type and provide the associated IP address on which the Receiver will listen.
2. Create a target whose transport type matches the transport protocol used for documents between Business Integration Connect and Message Broker.

For Community Manager to receive a document from Message Broker, it must know the location at which to retrieve the messages. This location must conform with the transport protocol to be used. The transport protocol must be one that Message Broker supports (see “Message transports that Message Broker supports” on page 130).

The following sections summarize how to create targets for transport protocols that Message Broker supports.

Configuring for incoming documents over HTTP transport protocol

When the Community Manager receives a document over the HTTP transport protocol, its Receiver retrieves the document from the defined target. This target identifies the URL at which the Receiver listens for documents from Message Broker. When Message Broker uses the HTTP transport protocol, the HTTPRequest node sends the document to the appropriate URL, where it can be received by the Community Manager.

For the Community Manager to receive documents through a target over the HTTP transport protocol, you must create a target from the Target List screen of the Community Console. This target must use the HTTP 1.1 transport protocol. The Community Manager determines this URL as is a combination of the following information:

- The IP address of the host machine, obtained from within the Community Manager’s participant profile
- The target URL, obtained from the URL field of the target definition

Note: An overview of how to create a target is provided in “Defining where to retrieve the back-end document” on page 31.

For Message Broker to be able to send documents to this target, the HTTPRequest node of the message flow must be configured to send documents to this URL. Therefore, you must ensure that this target URL is available to the Message Broker configuration.

Configuring for incoming documents over JMS transport protocol

When the Community Manager receives documents from Message Broker over the JMS protocol, the Community Manager obtains the document from the appropriate JMS input queue, where it has been transferred from the JMS output queue where Message Broker has sent it. For the Community Manager to be able to obtain this JMS location, you must create a target in Business Integration Connect, one that uses the JMS transport protocol. Through the target, the Community Manager listens for any documents on its input queue and retrieves them.

Note: For an overview of how to create a target, see “Defining where to retrieve the back-end document” on page 31.

For the Community Manager to receive documents through a target over the JMS transport, you must create a target from the Target List screen of the Community Console. When using WebSphere MQ version 5.3 as your JMS provider, use the information in Table 115 on page 178 to set the target fields. In addition, specify the information specified in Table 82 for the JMS protocol in the Target Details screen.

Table 82. JMS values for the Target Details screen for communication with Message Broker

Target Details field	Value	Notes and restrictions
JMS Queue Name	Name of the JMS input queue that receives documents from the output queue of Message Broker	Documents in this input queue are transferred from the JMS output queue on the machine where Message Broker resides

Configuring Message Broker

For your interactions between Business Integration Connect and Message Broker, you must create a Message flow project within the Broker Application Development Perspective of the Message Brokers Toolkit. This project will include the following artifacts:

- Message flows
- PIP files (RosettaNet only) or message definition files

Note: For more information on how to create message flow projects, see the WebSphere Business Integration Message Broker documentation set.

Creating the message flow

It is the **message flow**, within Message Broker, that performs the actual business logic you need to process information. Therefore, the appropriate message flows must exist for Message Broker to correctly process your Business Integration Connect documents. Make sure that a message flow exists that provides the business logic you need:

- If such a message flow does *not* currently exist, you must create or import one.
- If a message flow does exist, you must understand how to use it sufficiently to be able to use it.

For Message Broker to handle incoming and outgoing documents, its message flow uses special transport nodes. The type of transport node to use depends on the particular transport protocol, as shows.

Table 83. Creating message flows for different transport protocols

Transport protocol	For more information
HTTP	“Creating the message flow for HTTP transport” on page 136
HTTP (SOAP documents)	
JMS	“Creating the message flow for JMS transport” on page 144

Deploy the project

Once your message flow project contains the correct artifacts, you must deploy it to Message Broker. You deploy a message flow project with the Broker Administrator Perspective of the Message Brokers Toolkit.

Using HTTP transport protocol with Message Broker

This section describes how to send and receive documents between WebSphere Business Integration Connect and WebSphere Business Integration Message Broker through the use of the HTTP transport protocol.

Note: All references to the HTTP transport protocol apply to HTTPS as well.

Components required for documents over HTTP transport

To send or receive a document between Business Integration Connect and Message Broker using the HTTP transport protocol does not require any special components. Only Business Integration Connect and Message Broker are required. For Business Integration Connect to communicate with a version 5.0 of Message Broker using the HTTP transport protocol requires that these two components be configured. Table 84 summarizes these configuration steps.

Table 84. Configuring Business Integration Connect and Message Broker

Component	Version	For more information
WebSphere Business Integration Connect	4.2.2	“Configuring for outgoing documents over HTTP transport protocol” on page 132
		“Configuring for incoming documents over HTTP transport protocol” on page 133
WebSphere Business Integration Message Broker	5.0	“Configuring Message Broker” on page 134

In addition, to send or receive a document between Business Integration Connect and Message Broker using the HTTP transport protocol, you must use version 5.3 IBM WebSphere MQ as your JMS provider.

Sending documents over HTTP transport

For Business Integration Connect to send a document to Message Broker over the HTTP transport protocol, you use special HTTP-transport nodes within the Message Broker message flow to retrieve the document that Business Integration Connect has sent as an HTTP stream. The nodes of the message flow perform the computations required and then route the document to some destination (a JMS output queue).

The following steps describe how Business Integration Connect sends a document to a message flow within Message Broker over the HTTP transport protocol:

1. Business Integration Connect sends an HTTP message to Message Broker.
If the packaging type of the document was Backend Integration, Business Integration Connect has provided custom properties in this message.

Note: Within Business Integration Connect, you must configure a gateway that identifies the URL to which Business Integration Connect sends the message and on which Message Broker is polling. For more information, see “Configuring for outgoing documents over HTTP transport protocol” on page 132.

2. The HTTPInput node of the message flow picks up the document and sends it to the next node of the message flow. This node is usually a compute node.
3. The nodes of the message flow execute to perform the business logic.
When business logic is complete, the message flow sends the resulting document to its HTTPReply node.
4. The HTTPReply node, by default, sends back the output message to the client (Business Integration Connect).
Alternatively, the message flow can put the message into an MQOutput node. The MQOutput node receives the document and sends it to the appropriate JMS queue or other application.

Receiving documents over HTTP transport

For Business Integration Connect to receive a document from Message Broker using the HTTP transport protocol, you use special HTTP-transport nodes within the Message Broker message flow to send the document that Business Integration Connect is to receive as an HTTP stream. The nodes of the message flow perform the computations required and handle the request and response (if the interaction is synchronous) with Business Integration Connect.

The following steps describe how Business Integration Connect receives a document from a message flow within Message Broker over the HTTP transport protocol:

1. The message flow within Message Broker receives a document in its MQInput node (a JMS input queue).
2. The MQInput node of the message flow receives the document and sends it to the HTTPRequest node.
3. The HTTPRequest node handles the request and response interactions with the client (Business Integration Connect), using a specified URL.
4. Business Integration Connect receives the message from its URL, as configured in its target.

For more information on the target, see “Configuring for incoming documents over HTTP transport protocol” on page 133.

Creating the message flow for HTTP transport

For a Message Broker message flow to handle documents over the HTTP transport protocol, it uses the following transport nodes:

- HTTPInput
- HTTPReply
- HTTPRequest

The order of use for these transport nodes depends on the direction of communication, as follows:

- When Business Integration Connect *sends* a document to Message Broker, the message flow includes the types of nodes in Table 85 (in the order shown) to describe the business logic.
- When Business Integration Connect *receives* a document from Message Broker, the message flow includes the types of nodes in Table 86 (in the order shown) to describe the business logic.

Table 85. Nodes for sending documents to Message Broker over HTTP

Node type	Purpose	Notes
HTTPInput	Receives the Business Integration Connect request document into the message flow	Set this transport node's URL Selector field (in the Basic properties) to the URL where Business Integration Connect sends its documents. Therefore, it must be set to the URL configured in the Business Integration Connect gateway. The URL should have the following format: <i>http://hostName:port/path</i> where <i>hostName</i> is the name of the machine on which Message Broker resides, <i>port</i> is the HTTP port number on which the Message Broker is listening, and <i>path</i> identifies the location on this machine. For more information, see "Configuring for outgoing documents over HTTP transport protocol" on page 132.
Compute	Performs business-logic tasks, such as updating header information	Use ESQL to perform the business logic. The compute node sends the resulting message to the HTTPReply node.
HTTPReply	Returns a response to Business Integration Connect	By default, this node sends the output message to the client. However, you can configure it to send it to an MQOutput node.
MQOutput	Receives the document from the HTTPReply node and sends it to Business Integration Connect	This transport node sends the resulting document to a JMS output queue, which routes it to its next destination.

Table 86. Nodes for receiving documents from Message Broker over HTTP

Node type	Purpose	Notes
MQInput	Receives the document from Business Integration Connect	This transport node receives the incoming document from a JMS input queue.
HTTPRequest	Handles request/response interactions with Business Integration Connect	This transport node must set its Web Services URL field (in the Basic Properties) to the URL where Business Integration Connect is listening for documents. Therefore, it must be set to the URL configured in the Business Integration Connect target. The URL should have the following format: <i>http://hostName:port/bcgreceiver/path</i> where <i>hostName</i> is the name of the machine on which Business Integration Connect resides, <i>port</i> is the HTTP port number on which the Business Integration Connect Receiver is listening, and <i>path</i> identifies the location on this machine. For more information, see "Configuring for incoming documents over HTTP transport protocol" on page 133.

For more detailed information on how to create and configure message flow nodes, see your WebSphere Business Integration Message Broker documentation.

Sending SOAP documents

SOAP documents differ from other types of documents exchanged over HTTP/S. This section describes how to send and receive SOAP documents between WebSphere Business Integration Connect and WebSphere Business Integration Message Broker over the HTTP transport protocol.

The way to configure Business Integration Connect and Message Broker for transfer of SOAP documents is very similar to the configuration for transferring non-SOAP documents over the HTTP protocol. summarizes where to find information on how to configure these two integration components.

Table 87. Configuring Business Integration Connect and Message Broker for transfer of SOAP documents

Integration component	Configuration step	For more information
WebSphere Business Integration Connect	You configure the target and gateway the same way for SOAP documents as for non-SOAP documents over HTTP.	<p>“Configuring for outgoing documents over HTTP transport protocol” on page 132</p> <p>“Configuring for incoming documents over HTTP transport protocol” on page 133</p>
WebSphere Business Integration Message Broker	The message flows to handle SOAP documents are very similar to those for non-SOAP documents over HTTP. Only one additional transport node is required to handle SOAP documents.	<p>For sending a SOAP document to Message Broker, see Table 88.</p> <p>For receiving a SOAP document from Message Broker, see “Creating the message flow for HTTP transport” on page 136.</p>

For Message Broker to correctly process a SOAP document that Business Integration Connect sends, the message flow must contain an HTTPRequest node to handle communication with the Web Services client. Table 88 lists the nodes in a Message Broker message flow needed to handle a SOAP document sent by Business Integration Connect.

Table 88. Nodes for sending SOAP documents to Message Broker

Node type	Purpose	Notes
HTTPInput	Receives the Business Integration Connect request document into the message flow	<p>Set this transport node’s URL Selector field (in the Basic properties) to the URL where Business Integration Connect sends its documents. Therefore, it must be set to the URL configured in the Business Integration Connect gateway. The URL should have the following format:</p> <p><code>http://hostName:port/path</code></p> <p>where <i>hostName</i> is the name of the machine on which Business Integration Connect resides, <i>port</i> is the HTTP port number on which the Business Integration Connect Receiver is listening, and <i>path</i> identifies the location on this machine.</p> <p>For more information, see “Configuring for outgoing documents over HTTP transport protocol” on page 132.</p>

Table 88. Nodes for sending SOAP documents to Message Broker (continued)

Node type	Purpose	Notes
Compute	Performs business-logic tasks, such as updating header information	Use ESQL to perform the business logic. The compute node sends the resulting message to the HTTPReply node.
HTTPRequest	Sends the SOAP request to the external Web Service Provider (WebServices) and gets back a response from that WebService.	None
HTTPReply	Returns a response to Business Integration Connect	By default, this node sends the output message to the client.

Using JMS transport protocol with Message Broker

This section describes how to configure components to send and receive documents between WebSphere Business Integration Connect and WebSphere Business Integration Message Broker through the use of the JMS transport protocol. It provides the following information on how to send and receive documents:

- “Components required for documents over JMS transport”
- “Sending documents over JMS transport” on page 114
- “Receiving documents over JMS transport” on page 116

Components required for documents over JMS transport

To send or receive a document between Business Integration Connect and version 5.0 Message Broker using the JMS transport protocol requires WebSphere MQ as the JMS provider. The following sections describe how Business Integration Connect, Message Broker, and WebSphere MQ work together to exchange documents over the HTTP transport protocol.

Sending documents over JMS transport

For Business Integration Connect to send a document to Message Broker using the JMS transport protocol, you use the JMS queues. Business Integration Connect sends a document to its JMS output queue, where it is transferred to the JMS input queue on which Message Broker listens. When Message Broker receives a document, it retrieves it from its input queue. Message Broker’s message flow contains special WebSphere MQ (JMS) transport nodes, which handle access to the JMS queues. Figure 23 provides an overview of how Business Integration Connect sends documents to Message Broker over the JMS transport protocol.

WebSphere Business Integration Message Broker

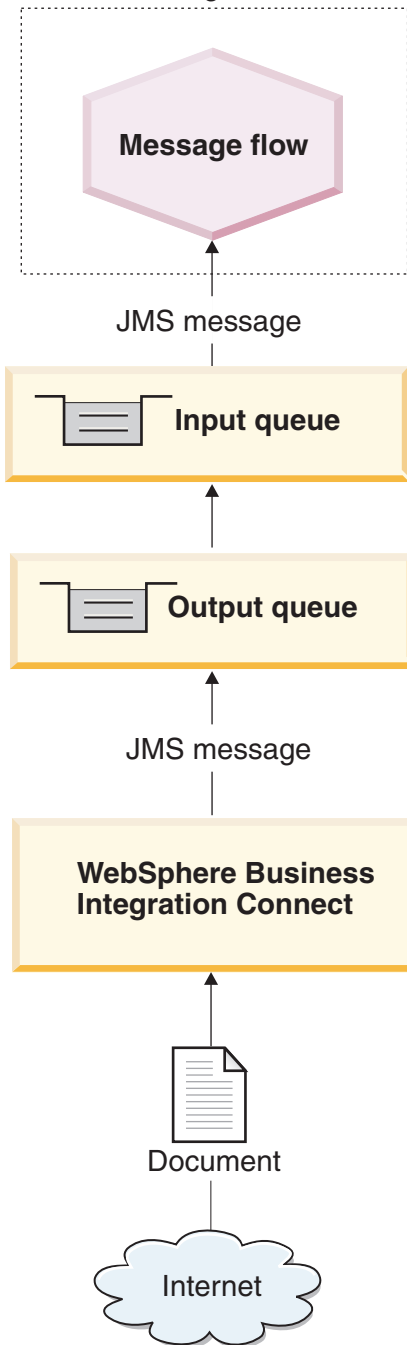


Figure 23. Message flow from Business Integration Connect to a message flow through the JMS transport protocol

The following steps describe how Business Integration Connect sends a document to a message flow within Message Broker over the JMS transport protocol:

1. Business Integration Connect posts a message to its JMS output queue.
If the packaging type of the document was Backend Integration, Business Integration Connect has provided custom properties in this message. The JMS message header, `JMSType`, is set with the content type of the payload.

Note: Within Business Integration Connect, you must configure a gateway that identifies the JMS output queue to which Business Integration Connect sends the message and on which Message Broker is polling. For more information, see “Configuring for outgoing documents over JMS transport protocol” on page 132.

2. WebSphere MQ transfers the document from the output queue on the machine where Business Integration Connect resides to the input queue that Message Broker is polling.
3. When Message Broker sees a message on its input queue, it retrieves the message and sends it to the appropriate message flow.
For information on how to set up this queue, see “Setting up the environment for JMS transport” on page 143.
4. The MQInput node sends the document to the next node of the message flow. This node is usually a compute node.
5. The nodes of the message flow execute to perform the business logic.
When business logic is complete, the message flow sends the resulting document to its MQOutput node.
6. The MQOutput node sends the document to the appropriate queue.

Receiving documents over JMS transport

For Business Integration Connect to receive a document from Message Broker over the JMS transport protocol, you use JMS queues. Message Broker sends a document to its JMS output queue, where it is transferred to the JMS input queue on which Business Integration Connect listens. When Business Integration Connect receives a document, it retrieves it from its input queue. Message Broker’s message flow contains special WebSphere MQ (JMS) transport nodes, which handle access to the JMS queues. Figure 24 provides an overview of how documents are sent from Message Broker to Business Integration Connect.

WebSphere Business Integration Message Broker

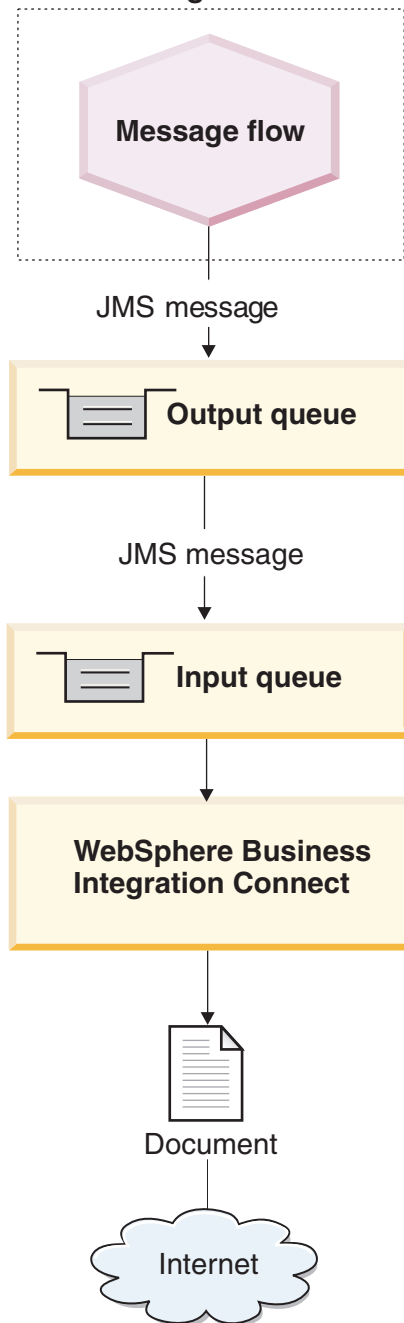


Figure 24. Message flow from a message flow to Business Integration Connect through the JMS transport protocol

The following steps describe how Business Integration Connect receives a document from a message flow within Message Broker over the JMS transport protocol:

1. The message flow within Message Broker receives a document in its MQInput node.
The message flow receives its incoming message from a JMS input queue.
2. The MQInput queue of the message flow receives the document and sends it to the next node of the message flow. This node is usually a compute node.

3. The nodes of the message flow execute to perform the business logic. When business logic is complete, the message flow sends the resulting document to its MQOutput node.
4. The MQOutput node sends the document to the appropriate JMS output queue.
5. WebSphere MQ transfers the document from the queue on the machine where Message Broker resides to the queue that Business Integration Connect is polling.
6. Business Integration Connect receives the message from its JMS input queue, as configured in its target.
For more information on the target, see “Configuring for incoming documents over JMS transport protocol” on page 134. For information on how to set up this queue, see “Setting up the environment for JMS transport.”

Setting up the environment for JMS transport

The sending and receiving of documents to and from Message Broker involves JMS queues (remote and local). For information on how to configure Business Integration Connect for use with Message Broker over JMS, see “Configuring Business Integration Connect for Message Broker” on page 131. To use the JMS transport protocol with Message Broker, you can set up the JMS system that WebSphere MQ provides. Version 5.0 of Message Broker uses version 5.3 of WebSphere MQ as a JMS provider. Therefore, you can use the steps in “Configuring a JMS protocol with WebSphere MQ,” on page 173 to set up the JMS transport-protocol mechanism.

Important: The steps in “Configuring a JMS protocol with WebSphere MQ,” on page 173 must be performed on the machine on which WebSphere Business Integration Connect resides. This guide assumes that the JMS transport-mechanism required by Message Broker has already been set up as part of the Message Broker installation.

When you create your JMS queues for use between Business Integration Connect and Message Broker, consider the following points:

- Part of the Message Broker installation process should involve the creation of the following queue managers:
 - A WebSphere MQ queue manager associated with the broker domain
You can use the following command to create this queue manager and a set of named queues:
`mqsicreatebroker`
 - A WebSphere MQ queue manager for Message Broker
Because Message Broker uses a set of predetermined queue names, it requires a separate WebSphere MQ queue manager per broker. Message Broker can share this queue manager hosting with either its Configuration Manager or the optional User Name Server, or both.

For more information, refer to your *WebSphere Business Integration Message Broker Installation and Configuration Guide*.

- When you create your JMS queue aliases, you might want to name them to indicate the direction of flow between Business Integration Connect and Message Broker.

For example, if you create the queues listed in the Original queue name column of Table 73, you could rename these queues to indicate the Message Broker directionality, as shown in the Directional queue name column of Table 89.

Table 89. Naming JMS queues for Message Broker directionality

Original queue name	Directional queue name
inQ	MB2WBIC
outQ	WBIC2MB

Creating the message flow for JMS transport

For a Message Broker message flow to handle documents over the JMS transport protocol, it uses the following transport nodes:

- MQInput
- MQOutput

The order of use for these transport nodes depends on the direction of communication, as follows:

- When Business Integration Connect *sends* a document to Message Broker, the message flow includes the types of nodes in Table 90 (in the order shown) to describe the business logic.
- When Business Integration Connect *receives* a document from Message Broker, the message flow includes the types of nodes in Table 91 (in the order shown) to describe the business logic.

Table 90. Nodes for sending documents to Message Broker over JMS

Node type	Purpose	Notes and restrictions
MQInput	Receives the document from Business Integration Connect	The value in the Queue Name field (in the Basic properties) of this transport node is the message flow's input queue. WebSphere MQ must be set up so that this JMS queue receives documents from Business Integration Connect's output queue. For more information, see "Configuring for outgoing documents over JMS transport protocol" on page 132.
Compute	Performs business-logic tasks, such as removing header information	None
MQOutput	Receives the document from the compute node and sends it as the message-flow output	This transport node sends the resulting document to a JMS output queue, which routes it to its next destination.

Table 91. Nodes for receiving documents from Message Broker

Node type	Purpose	Notes and restrictions
MQInput	Receives the document into the message flow	This transport node receives the incoming document from a JMS input queue.
Compute	Performs business-logic tasks, such as updating header information	None
MQOutput	Receives the document from the compute node and sends it to Business Integration Connect	The value in the Queue Name field (in the Basic properties) of this transport node is the message flow's output queue. WebSphere MQ must be set up so that this JMS queue sends documents to Business Integration Connect's input queue. For more information, see "Configuring for outgoing documents over HTTP transport protocol" on page 132.

For more detailed information on how to create and configure message flow nodes, see your WebSphere Business Integration Message Broker documentation.

Chapter 6. Integrating with WebSphere Data Interchange

This chapter describes how to integrate WebSphere Business Integration Connect with the WebSphere Data Interchange.

Note: For a description of the general process used to integrate Business Integration Connect with a back-end system, see Chapter 1, “Planning for back-end integration,” on page 3.

This chapter provides an explanation of the process by which documents are exchanged and then lists the steps for setting up a sample environment for such exchanges. The scenario used throughout this chapter is similar to the one presented in the *Integrating WebSphere Data Interchange V3.2 with WebSphere Business Integration Connect V4.2* tutorial, which is available on the following Web site:

www.ibm.com/developerworks/websphere/

The tutorial provides additional scripts (in the section on configuring WebSphere MQ) as well as sample transformation maps. By following the tutorial, you can set up the environment described in this chapter.

It is assumed that you are familiar with using WebSphere Data Interchange. See the WebSphere Data Interchange documentation for additional information as you read this chapter.

Introduction

WebSphere Data Interchange integrates electronic data interchange (EDI) into the WebSphere business process, messaging, and Internet-based B2B capabilities. You exchange documents and messages between Business Integration Connect and WebSphere Data Interchange through the JMS transport protocol. You must specify a packaging of None when sending a document to WebSphere Data Interchange.

Note: WebSphere Data Interchange provides other types of integration options, such as file-based integration. Refer to the WebSphere Data Interchange documentation for details on enabling the exchange of documents through file-based integration.

Sending documents to WebSphere Data Interchange

For Business Integration Connect to send an EDI document to WebSphere Data Interchange, the following steps occur:

1. A community participant sends an EDI document to Business Integration Connect. The document is sent through the AS2 over HTTP transport protocol. Business Integration Connect strips off the AS2 packaging from the EDI document.
2. Business Integration Connect places the EDI document on a queue.

Note: Business Integration Connect determines the protocol used in the document by examining the first three characters of the EDI document. It then determines, from the protocol type, the sender and receiver information. See “Overview of EDI routing” on page 167 for details.

- WebSphere Data Interchange reads the EDI document from the queue. It performs the tasks of unwrapping, validating, and translating the EDI document.
- Note:** WebSphere Data Interchange must be configured for user profiles and the desired mappings.
- WebSphere Data Interchange distributes the document to a back-end system. If the back-end system is WebSphere InterChange Server, WebSphere Data Interchange sends the document to the WebSphere Business Integration Adapter for MQ to create a business object and invoke a collaboration within InterChange Server.

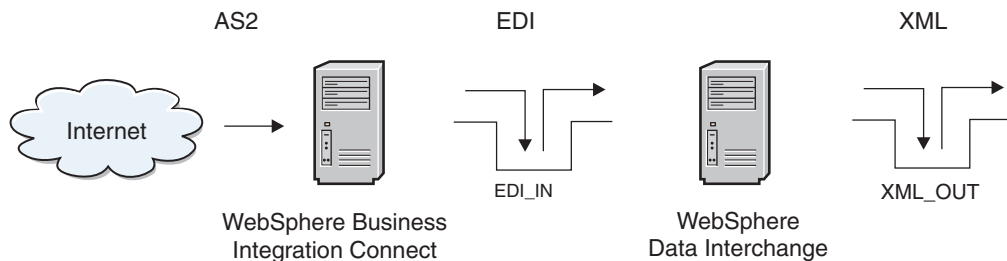


Figure 25. EDI document from Business Integration Connect

In Figure 25, a community participant sends an AS2 document to Business Integration Connect, which, in turn, sends it to the EDI_IN queue on the WebSphere Data Interchange side. Note that the remote queue, transmission queue, receiver queue (in the example, EDI_IN), and the sender and receiver channels must be set up so that the message sent to Business Integration Connect is transmitted to the EDI_IN queue. The WebSphere Data Interchange server picks up the EDI document, searches for the user profiles, mappings, and so on, converts the document to XML, and puts it in the XML_OUT queue.

Receiving documents from WebSphere Data Interchange

For Business Integration Connect to receive an EDI document from WebSphere Data Interchange, the following steps occur:

- WebSphere Data Interchange places the EDI document on a queue.
- Business Integration Connect reads the message from the queue.

Note: Business Integration Connect determines how to route the document as described in “Overview of EDI routing” on page 167.

- Business Integration Connect routes the document to the appropriate community participant.

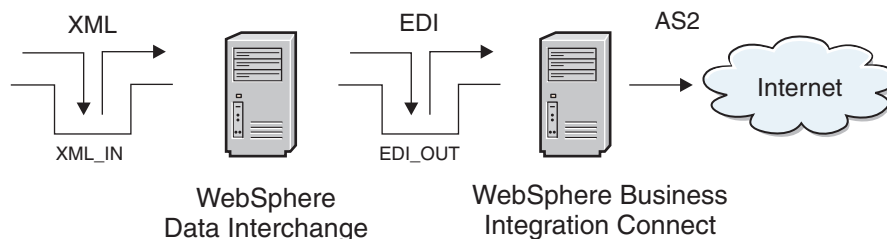


Figure 26. Sending an EDI document to WebSphere Business Integration Connect

In Figure 26, an XML document is placed into the XML_IN queue for WebSphere Data Interchange to translate. It is assumed that the user profiles, mappings, and so on, are already performed. Upon receiving a valid XML document, WebSphere Data Interchange converts it into EDI format and places the output in the EDI_OUT queue (a remote queue). It is also assumed that the transmission queue, sender and receiver channels, and receiver queue on the Business Integration Connect side are set up. Upon receiving the document, Business Integration Connect routes it to the community participant.

Sample scenario used in this chapter

Throughout this chapter, you will see the steps to set up the exchange of EDI documents between two trading partners. The EDI documents are sent over the internet, and AS2 (over HTTP) is used as the communication protocol.

In this sample, the trading partners are Partner One and Partner Two. Figure 27 illustrates the configurations of the two partners.

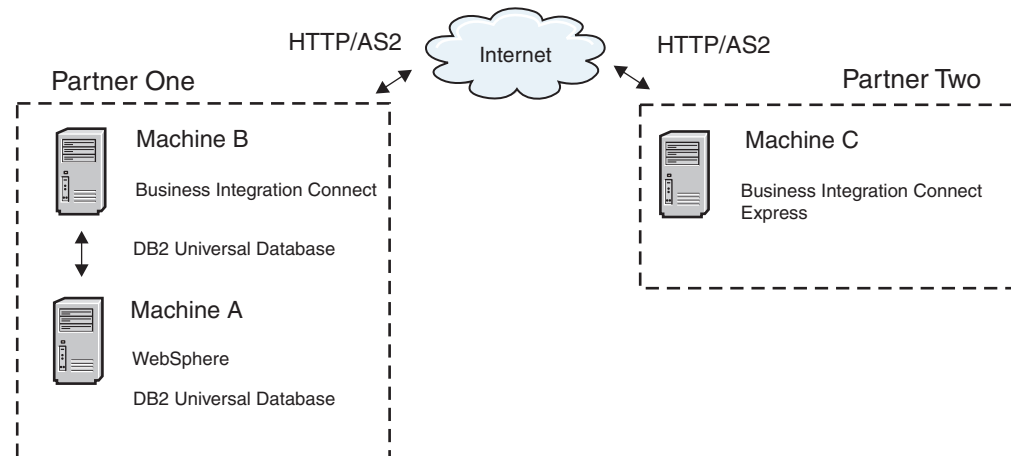


Figure 27. Configuration of two partners in sample scenario

The following software is used to implement this sample scenario:

- On Machine A (Partner One):
 - Operating System: Microsoft Windows 2000 Professional
 - WebSphere Data Interchange Server V3.2 with CSD 07 (or higher)
 - WebSphere Data Interchange Client V3.2 with Fix Pack 7 (or higher)
 - WebSphere MQ V5.3 with CSD 04
 - IBM DB2 V7.2 with Fix Pack 10
- On Machine B (Partner One):
 - Operating System: Red Hat Linux Advanced Server v2.1
 - WebSphere Business Integration Connect Enterprise Edition v4.2.0 (or higher)
 - WebSphere MQ V5.3 with CSD 04
 - IBM DB2 V8.1 with Fix Pack 2
- On Machine C (Partner Two):
 - Operating System: Windows 2000 Professional
 - WebSphere Business Integration Connect Express v4.2.0 (or higher)

Refer to the *Business Integration Connect Installation Guide* and to the WebSphere Data Interchange documentation for a complete list of software prerequisites.

In this example, partnerOne is operating two machines. Machine A has both WebSphere MQ and WebSphere Data Interchange Server installed. Machine B has WebSphere MQ as well as WebSphere Business Integration Connect Enterprise Edition installed. Machine B supports the communications between the two trading partners.

WebSphere Data Interchange supports integration with WebSphere MQ, enabling interoperability with a wide range of enterprise applications and business process engines. WebSphere Business Integration Connect employs WebSphere MQ as a JMS provider. As such, integration between WebSphere Data Interchange and WebSphere Business Integration Connect is through MQ messages destined for JMS API clients.

WebSphere Business Integration Connect is used to communicate EDI transactions over the Internet using the AS2 protocol.

Note that, in this example, partnerTwo is using WebSphere Business Integration Connect - Express to accept transactions via AS2 and has its own WebSphere Data Interchange environment for handling translations and acknowledgments.

Throughout this chapter, you will see the details about configuring the machines used in this sample scenario. The flow of messages is bi-directional, and so both send and receive artifacts are included.

Configuring your environment for message exchange

To enable communication between WebSphere Data Interchange and Business Integration Connect, you perform the following setup and configuration tasks:

- “Configure WebSphere MQ communication”
- “Configuring WebSphere Data Interchange” on page 152
- “Setting up the JMS environment” on page 156
- “Configuring Business Integration Connect Enterprise Edition” on page 157

Configure WebSphere MQ communication

The first step in setting up the environment is to configure WebSphere MQ intercommunication. Intercommunication means sending messages from one queue manager to another. The first step is to define a queue manager (and associated objects) for the WebSphere Data Interchange system and the Business Integration Connect system. If you will be sending messages in both directions, you set up a source queue manager and a target queue manager on both systems. On the source queue manager, you define a sender channel, a remote queue definition, and a transmission queue. On the target queue manager, you define a receiver channel and a target queue.

Note: Refer to the WebSphere MQ documentation for additional details on defining queue managers.

This section shows you the values you would use to set up the queue managers and associated objects needed for the sample scenario. In the scenario, WebSphere MQ V5.3 is installed on both Machine A and Machine B. The first step, then, is to create a queue manager on both Machine A and Machine B for use with WebSphere Data Interchange and WebSphere Business Integration Connect Enterprise Edition respectively.

Note: Your WebSphere Data Interchange queue manager should be configured to trigger the WebSphere Data Interchange Server using the WDI Adapter application.

- On Machine A, you would use the queue manager defined for use with WebSphere Data Interchange. For the remainder of this chapter, this queue manager is referred to as WDI32_QM.
- On Machine B, you would use the queue manager created during the initial installation and configuration of WebSphere Business Integration Connect Enterprise Edition. For the remainder of this chapter, this queue manager is referred to as WBIC42_QM

To send messages from one queue manager to another using WebSphere MQ, you define the following objects:

- On the source queue manager:
 - Sender channel
 - Remote queue definition
 - Transmission queue
- On the target queue manager:
 - Receiver channel
 - Target queue

In the sample scenario, both Machine A and Machine B act as sender and receiver. Therefore, you would have to define a number of objects on each machine.

Table 92 lists the objects you would create to set Machine A and Machine B as sender and receiver.

Table 92. WebSphere MQ objects to create

WebSphere MQ object	Machine A	Machine B
Queue Manager	WDI32_QM	WBIC42_QM
Sender Channel	TO.WBIC42	TO.WDI32
Receiver Channel	TO.WDI32	TO.WBIC42
Remote Queue	EDI_OUT_A	EDI_OUT_B
Transmission Queue	XMITQ_A	XMITQ_B
Local Queue	EDI_IN_A	EDI_IN_B
Local Queue	XML_IN_A	XML_IN_B
Local Queue	XML_OUT_A	XML_OUT_B

Figure 28 shows the message flow between Machine A and Machine B, indicating the role of the WebSphere MQ objects listed in Table 92.

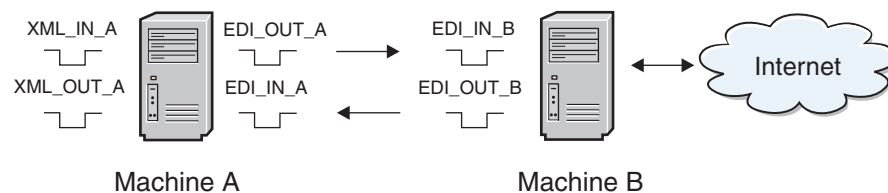


Figure 28. Message flow between Machine A and Machine B

You could use several different methods to define these objects, depending on your WebSphere MQ platform. For example, you could use WebSphere MQ Explorer on Windows to define the objects.

Configuring WebSphere Data Interchange

For WebSphere Data Interchange to receive messages from the WebSphere MQ queue and write EDI messages to a queue, you must configure profiles in the WebSphere Data Interchange Client. Using WebSphere Data Interchange Client, you would create the following profiles, which are described in the sections that follow:

- MQ Series queue profile
- Network profile
- Mailbox profile
- Service profile

In the sample scenario, WebSphere Data Interchange receives XML messages from the WebSphere MQ queue XML_IN_A and writes the result of translation to WebSphere MQ queue EDI_OUT_A. This is called the XML-to-EDI translation. WebSphere Data Interchange also receives EDI from WebSphere Business Integration Connect Enterprise Edition on the WebSphere MQ queue EDI_IN_A and writes the result of translation to XML_OUT_A.

MQSeries Queue profile

An MQSeries Queue profile contains information about a WebSphere MQ message queue. Table 93 shows the properties to configure for each profile.

Table 93. Properties in an MQSeries Queue profile

MQ property	Description
Queue Profile ID	The unique identifier to name the profile (logical name)
Full Queue Name	The actual name of the WebSphere MQ queue
Queue Manager Name	The actual name of the WebSphere MQ queue manager
Description	Any string to identify the purpose of the profile
Maximum Length	The largest possible message for the queue as configured in WebSphere MQ
Destructive Reads	If selected, these cause WebSphere Data Interchange to remove the message from the WebSphere MQ queue when reading.
Syncpoint Control	When checked, the reading and writing of queue messages is under syncpoint control. If syncpoint control is in effect, modifications to a message queue do not take place until WebSphere Data Interchange issues a syncpoint.

Because you're working with the WebSphere MQ queues, you require an MQSeries Queue profile in WebSphere Data Interchange for *each* queue. In all, you would create four MQSeries Queue profiles, one for each WebSphere MQ queue used in the message flow. From the setup area of WebSphere Data Interchange Client, you would:

1. Create an MQSeries Queue profile for XML_IN_A and EDI_OUT_A.

Table 94 lists the actual parameters specified in each MQSeries Queue profile you created. The queues represented here are used with XML-to-EDI translation.

Table 94. MQSeries Queue profile for XML_IN_A and EDI_OU_A

Queue property	Value for XML_IN_A	Value for EDI_OU_A
Queue Profile ID	XML_IN_A	EDI_OU_A
Full Queue Name	XML_IN_A	EDI_OUT_A
Queue Manager Name	WDI32_QM	WDI32_QM
Destructive Reads	Checked	Checked
Syncpoint Control	Checked	Checked

Note: The Queue Profile ID is restricted to a maximum of eight characters. Therefore, the profile ID for the EDI_OUT_A queue must be named EDI_OU_A. All references to the WebSphere MQ queue EDI_OUT_A in WebSphere Data Interchange use EDI_OU_A.

2. Create an MQSeries Queue profile for EDI_IN_A and XML_OU_A. Table 95 defines the properties of each queue used in EDI-to-XML translation.

Table 95. MQSeries Queue profile for EDI_IN_A and XML_OU_A

Queue property	Value for EDI_IN_A	Value for XML_OU_A
Queue Profile ID	EDI_IN_A	XML_OU_A
Full Queue Name	EDI_IN_A	XML_OUT_A
Queue Manager Name	WDI32_QM	WDI32_QM
Destructive Reads	Checked	Checked
Syncpoint Control	Checked	Checked

Network profile

Network profiles define for WebSphere Data Interchange the characteristics of the networks you use for communications with trading partners. For this scenario, you would create and configure a Network Profile that communicates with the WebSphere MQ queues created earlier.

Table 96 shows the properties to configure for the Network profile.

Table 96. Properties in a Network profile

Network property	Description
Network ID	A unique identifier to name the profile
Communication Routine	The name of the program that builds network commands and invokes the network program to process the commands
Network Program	The program invoked by the communication routine to process requests
Network Parameters	Parameters required by the network program

For this sample scenario, you create and configure a Network profile that communicates with the WebSphere MQ queues created earlier (see “MQSeries Queue profile” on page 152), as follows:

1. Create a new Network profile called WBIC_IN.

This network profile is used in the XML-to-EDI scenario. Table 97 lists the actual parameters specified for WBIC_IN.

Table 97. Network profile for WBIC_IN

Network property	Value for WBIC_IN profile
Network ID	WBIC_IN
Communication Routine	VANIMQ
Network Program	EDIMQSR
Network Parameters	SENDMQ=EDI_OU_A RECEIVEMQ=XML_IN_A

2. Create a second Network profile called WBIC_OUT.

This Network profile is used in the translation of EDI received from WebSphere Business Integration Connect Enterprise Edition. A second Network profile is required, because WebSphere Business Integration Connect Enterprise Edition places messages on the WebSphere MQ queues that include RFH2 headers.

Table 98 lists the properties of WBIC_OUT.

Table 98. Network profile for WBIC_OUT

Network property	Value for WBIC_OUT profile
Network ID	WBIC_OUT
Communication Routine	VANIMQ
Network Program	EDIRFH2
Network Parameters	SENDMQ=XML_OU_A RECEIVEMQ=EDI_IN_A

Mailbox profile

Mailbox profiles contain the information that WebSphere Data Interchange needs to identify the individuals and groups in your organization that receive documents to be translated. Table 99 shows the properties to configure for each Mailbox profile.

Table 99. Properties in a Mailbox profile

Mailbox property	Description
Mailbox ID	A unique identifier to name the profile
Network ID	The network ID of the network profile created earlier

You create mailbox profiles for each of the WebSphere MQ queues to identify the individuals and groups in the organization, as follows:

1. Create a Mailbox profile for each WebSphere MQ queue used.

Table 100. lists the actual parameters in each of the Mailbox profiles.

Table 100. Mailbox profiles for XML_IN_A and EDI_OU_A

Mailbox property	Value for XML_IN_A	Value for EDI_OU_A
Mailbox ID	XML_IN_A	EDI_OU_A
Network ID	WBIC_IN	WBIC_IN
Receive File	XML_IN_A	EDI_OU_A

2. Create a second pair of mailboxes.

Table 101 lists the properties for each.

Table 101. Mailbox profiles for EDI_IN_A and XML_OU_A

Mailbox property	Value for EDI_IN_A	Value for XML_OU_A
Mailbox ID	EDI_IN_A	XML_OU_A
Network ID	WBIC_OUT	WBIC_OUT
Receive File	EDI_IN_A	XML_OU_A

Service profile

Service profiles allow you to enter a utility command and define all the files that will be used during execution of that command.

For the sample scenario, you take the following steps:

1. Create a new Service Profile for XML_IN_A. You define the properties under the **General** tab, as follows:

- Continue Command Chaining: **On Success**
- PERFORM Command:

```
PERFORM TRANSFORM WHERE INFILE(XML_IN_A) SYNTAX(X)
OUTTYPE(MQ)OUTFILE(EDI_OU_A)
```

Table 102 lists the Common Files properties.

Table 102. Common Files for XML_IN_A

Common File property	Value
Tracking File	..\trk\xml_in.trk
Exception File	..\xex\xml_in.xex
Work File	..\wrk\xml_in.wrk
Report File	..\rpt\xml_in.rpt
Query File	..\qry\xml_in.qry

2. Enter the following in the **Output Files** tab:

- Name in Command: **EDI_OU_A**
- System File Name: **..\edi\edi_out.txt**

Note: EDI_OU_A is used rather than EDI_OUT_A because of character length restrictions.

3. Create a second Service Profile for EDI_IN_A. You define the properties under the **General** tab, as follows:

- Continue Command Chaining: **On Success**
- PERFORM Command:

```
PERFORM TRANSFORM WHERE INFILE(EDI_IN_A) SYNTAX(E)
OUTTYPE(MQ) OUTFILE(XML_OU_A)
```

Table 103 lists the Common Files properties.

Table 103. Common files for EDI_IN_A

Common File property	Value
Tracking File	..\trk\edi_in.trk
Exception File	..\xex\edi_in.xex
Work File	..\wrk\edi_in.wrk
Report File	..\rpt\edi_in.rpt
Query File	..\qry\edi_in.qry

4. Enter the following details under the **Output Files** tab:

- Name in Command: **XML_OU_A**
- System File Name: **..\xml\xml_out.txt**

Note: XML_OU_A is used rather than XML_OUT _A because of character length restrictions. This restriction was eliminated with CSD10 of the WebSphere Interchange Server.

Import and compile data transformation maps

After you create the profiles, as described in the previous section, you can import any maps you need to transform your data. You then compile the transformation maps and set a rule for each. You use the WebSphere Data Interchange Client to perform these tasks. See the WebSphere Data Interchange documentation for information.

Setting up the JMS environment

As mentioned earlier in this chapter, WebSphere Business Integration Connect Enterprise Edition can use the WebSphere MQ implementation of the Java Message Service (JMS) for integration with WebSphere Data Interchange.

Note: Alternatively, it is possible to use LDAP or WebSphere Application Server as a JNDI provider.

This section outlines the steps involved in creating a JMS environment on Machine B:

- “Configuring JMSAdmin”
- “Creating the JMS objects” on page 157

WebSphere MQ classes for Java and WebSphere MQ classes for JMS are built in to WebSphere MQ for Windows version 5.3.

Configuring JMSAdmin

Use the JMSAdmin tool available in WebSphere MQ to create the JMS objects in JNDI. For information on how to create the default configuration file called `JMSAdmin.config`, see “Configuring a JMS protocol with WebSphere MQ,” on page 173.

To create the JMS objects for this tutorial:

1. To use a file-based JNDI provider, you would make sure the `JMSAdmin.config` file contains the lines shown below:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.ReffsContextFactory
PROVIDER_URL=file:/opt/mqm/java/JNDI
```
2. If the JNDI directory does not already exist, create the JNDI directory under the following directory:
`/opt/mqm/java/bin`

Before invoking the JMSAdmin tool, you would ensure your CLASSPATH contains the following entries:

```
/opt/mqm/java/lib/jms.jar
/opt/mqm/java/lib/com.ibm.mq.jar
/opt/mqm/java/lib/com.ibm.mqjms.jar
/opt/mqm/java/lib/jta.jar
```



```
/opt/mqm/java/lib/connector.jar
/opt/mqm/java/lib/jndi.jar
/opt/mqm/java/lib/providerutil.jar
/opt/mqm/java/lib/fscontext.jar
```

Note: The above entries, which relate to Linux, assume you are using file-based JNDI.

Creating the JMS objects

To create the required JMS objects, you use the JMSAdmin tool. For the sample scenario, you would:

1. Define a new context:
DEF CTX(WdiJms)
2. Change to the new context:
CHG CTX(WdiJms)
3. Define a queue connection factory:
DEF QCF(WBIC42_QM_QCF) TRAN(CLIENT) HOST(IP_MACHINE_B)
PORT(9999) CHAN(java.channel) QMANAGER(WBIC42_QM)
4. Define the EDI_IN_B queue:
DEF Q(EDI_IN_B) QMANAGER(WBIC42_QM) QUEUE(EDI_IN_B)
5. Define the EDI_OUT_B queue:
DEF Q(EDI_OUT_B) QMANAGER(WBIC42_QM) QUEUE(EDI_OUT_B)
6. End the JMSAdmin session
END

Configuring Business Integration Connect Enterprise Edition

WebSphere Business Integration Connect is the communication layer between disparate community participants and internal processes. When setting up Business Integration Connect to work with EDI documents, you can configure it to:

- Send and receive EDI documents to and from WebSphere Data Interchange
- Communicate EDI transactions with external trading partners using AS2

The *Hub Configuration Guide* provides complete information on how to configure WebSphere Business Integration Connect Enterprise and Advanced editions. This section provides you with an example of configuring the WebSphere Business Integration Connect Enterprise Edition that is described in the sample scenario. It describes the following steps:

1. “Creating participants”
2. “Setting the B2B capabilities” on page 159
3. “Creating gateways” on page 160
4. “Defining document flow definitions and interactions” on page 162
5. “Creating participant connections” on page 162
6. “Creating targets” on page 163

Note: For information on how to configure WebSphere Business Integration Connect - Express, see “Configuring Business Integration Connect - Express” on page 164.

Creating participants

A participant profile identifies companies to the system. You create participants for Partner One and Partner Two in the WebSphere Business Integration Connect Enterprise Edition Community Console.

Create a participant for Partner One: Create a participant profile to represent Machine A and Machine B, which are the two systems that Partner One owns.

To create this participant profile, you take the following steps:

1. Open the WebSphere Business Integration Connect Community Console.
2. Log in as the **Hub Operator**.
3. Verify that **Profiles** is already selected from the Account Admin menu.
4. Click **Create** and enter the details as listed in Table 104 below.

Table 104. Participant properties for Partner One

Field name	Value
Participant Login Name	partnerOne
Participant Name	Partner One
Participant Type	Community Manager
Status	Enabled
Vendor Type	Other
Web Site	http://IP_MACHINE_A
	where IP_MACHINE_A is the Internet protocol (IP) address of Machine A
Business ID Type	Freeform
Business ID Identifier	123456789
IP Address Gateway Type	Production
IP Address	IP_MACHINE_A
	where IP_MACHINE_A is the Internet protocol (IP) address of Machine A

Note: To create the Business ID Type and Business ID Identifier, you first click on the **New** button below Business ID. The Business ID must be unique. Similarly, to create details relating to the IP Address, you click on the **New** button below the IP Address header.

5. Click **Save**.

WebSphere Business Integration Connect Enterprise Edition uses the Business ID Identifier (defined in Table 104) to identify the sender or receiver of a document. When an ANSI X12 EDI transaction is received, the Interchange Sender and Receiver data is read to determine the source and target of the transaction.

Important: Make a note of the Administrator's Password for Partner One, as you will need it later. When you logged on to the Community Console as Partner One, you were asked to enter the password and then to change it.

Create a participant for Partner Two: Next, create a community participant to represent Partner Two. To create the participant, you take the following steps:

1. Click **Account Admin** on the main menu and **Profiles** on the horizontal navigation bar.
2. Click **Create**.

3. Enter the values listed in Table 105 below.

Table 105. Participant properties for Partner Two

Field name	Value
Participant Login Name	partnerTwo
Participant Name	Partner Two
Participant Type	Community Participant
Status	Enabled
Vendor Type	Other
Web Site	http://IP_MACHINE_C
	where IP_MACHINE_C is the Internet protocol (IP) address of Machine C
Business ID Type	Freeform
Business ID Identifier	987654321
IP Address Gateway Type	Production
IP Address	IP_MACHINE_C
	where IP_MACHINE_C is the Internet protocol (IP) address of Machine C


4. Click **Save**.


Important: Make a note of the Administrator's Password for Partner Two, as you will need it later. When you logged on to the Community Console as Partner Two, you were asked to enter the password and then to change it.

Setting the B2B capabilities




You define the B2B capabilities for each participant in WebSphere Business Integration Connect Enterprise Edition through the Community Console. After you define the B2B capabilities for participants, you can define a valid Document Flow Definition used to support specific business collaboration types between the participants.


Set the B2B capabilities for Partner One: To define the B2B Capabilities for Partner One, take the following steps:

1. Click **Account Admin** on the main menu and **Profiles** on the horizontal navigation bar.
2. Click **Search** to reveal a list of all participants defined in the system.
3. Click the  icon next to **Partner One**, and then click **B2B Capabilities**.


B2B Capabilities are set to active by clicking on the  icon. For the purposes of this sample, you will configure only the B2B Capabilities required to implement the scenario.

To set the source and target packaging for Partner One to None, you would:






1. Click the  icon underneath **Set Source for Package: None** to enable it. Repeat this step for **Set Target**.
2. Click the  icon to drill down.
3. Click the  icon for **Protocol: EDI-X12 (ALL)** for both source and target.

4. Click the  icon.
5. Click the  icon for **Document Flow: All** for both source and target.


Set the B2B capabilities for Partner Two: To define the B2B capabilities for Partner Two, take the following steps:

1. Click **Account Admin** on the main menu and **Profiles** on the horizontal navigation bar.
2. Click **Search** to reveal a list of all participants defined in the system.
3. Click the  icon next to **Partner Two**, and then click **B2B Capabilities**.

To set the source and target packaging for Partner Two to AS, take the following steps:

1. Click the  icon underneath **Set Source for Package: AS** to enable it. Repeat this step for **Set Target**.
2. Click the  icon to drill down.
3. Click the  icon for **Protocol: EDI-X12 (ALL)** for both source and target.
4. Click the  icon.
5. Click the  icon for **Document Flow: All** for both source and target.

Next, you update the AS definition for Partner Two, to ensure that Message Disposition Notifications (MDNs) for AS2 sent to Partner Two are returned to the correct address, as follows:

1. Click on the **Edit** icon ().
2. Enter an AS MDN E-mail address.
This address is used to receive MDNs for AS1.
3. Enter an AS MDN HTTP URL:
`http://IP_MACHINE_B:PORT/bcgreceiver/submit`

Note: The URL defined for AS2 uses the same parameters that will be defined for the AS2 Target later in this chapter.

Creating gateways

A gateway in Business Integration Connect defines a network point that acts as the entrance to another network. The gateway contains the information that tells WebSphere Business Integration Connect how to deliver documents to the Enterprise Application Integration (EAI) layer.

Create a Gateway for Partner One: Partner Two sends EDI documents to Partner One using AS2. Partner One's gateway is used to send the EDI documents received via AS2 to a JMS queue and ultimately to WebSphere Data Interchange for translation.

To create a new gateway for Partner One, take the following steps:

1. Click **Account Admin** from the main menu and **Profiles** from the horizontal navigation bar.


2. Click **Search**.
3. Select Partner One by clicking the  icon, and then select **Gateways**.
4. Click **Create** to create a new gateway for Partner One.
5. Enter the values for this new gateway are shown in Table 106.

Table 106. Properties for Partner One gateway

Field name	Value
Gateway Name	JMStoPartnerOne
Transport	JMS
Target URI	file:///opt/mqm/java/JNDI/WdiJms
JMS Factory Name	WBIC42_QM_QCF
JMS Message Class	TextMessage
JMS Message Type	TextMessage
JMS Queue Name	EDI_OUT_B
JMS JNDI Factory Name	com.sun.jndi.fscontext.RefFSContextFactory

6. Click **Save**.

Make JMStoPartnerOne the default gateway for Partner One, as follows:

1. Click **View Default Gateways**.
2. From the **Production** list, select **JMS2toPartnerOne**.
3. Click **Save**.

Note: A JMS Gateway can be defined only for the Community Manager (Partner One, in the sample scenario).

Create a Gateway for Partner Two: Partner One sends EDI documents to WebSphere Business Integration Connect Enterprise Edition over a JMS queue. Partner Two's gateway is used to send the received EDI documents to Partner Two via AS2.

To create a new gateway for Partner Two, take the following steps:


1. Click **Account Admin** from the main menu and **Profiles** from the horizontal navigation bar.
2. Click **Search**.
3. Select Partner Two by clicking the  icon, and then select **Gateways**.
4. Click **Create** to create a new gateway for Partner Two.
5. Enter the values for this gateway as shown in Table 107.

Table 107. Properties for Partner Two gateway

Gateway Name	AS2toPartnerTwo
Transport	HTTP/1.1
Target URI	http://IP_MACHINE_C/input/AS2
User Name	partnerOne
Password	partnerOne

6. Click **Save**.

Note: The User Name and Password as entered above refer to the Inbound Participant Mapping Method for HTTP as defined in WebSphere Business Integration Connect - Express.

For an example of setting these properties in WebSphere Business Integration Connect - Express, see “Configuring Business Integration Connect Enterprise Edition” on page 157.

Notice that AS2toPartnerTwo is displayed as Online with a Status of **Enabled**.

Make AS2toPartnerTwo the default gateway for PartnerTwo, with the following steps:

1. Click **View Default Gateways**.
2. From the **Production** list, select **AS2toPartnerTwo**.
3. Click **Save**.

Defining document flow definitions and interactions

A document flow definition is a collection of “meta-information” that defines the document processing capabilities of the participant. For the system to process a business document, two or more document flow definitions must be linked to create an interaction.

To create a document flow definition and a valid interaction between Partner One and Partner Two, take the following steps:

1. Click **Hub Admin** from the main menu and **Document Flow Definition** from the horizontal navigation bar.
2. Click **Manage Interactions** and then **Create a Valid Interaction**.
3. From the Source column, select:
 - a. Package: **None**
 - b. Protocol: **EDI-X12**
 - c. Document Flow: **All**
4. From the Target column, select:
 - a. Package: **AS**
 - b. Protocol: **EDI-X12**
 - c. Document Flow: **ALL**
5. Set the Action as **Pass Through**.
6. Click **Save**.
7. Click **Create a Valid Interaction** again.
8. From the Source column, select:
 - a. Package: **AS**
 - b. Protocol: **EDI-X12**
 - c. Document Flow: **ALL**
9. From the Target column select:
 - a. Package: **None**
 - b. Protocol: **EDI-X12**
 - c. Document Flow: **All**
10. Set the Action as **Pass Through**.
11. Click **Save**.

Creating participant connections

Participant connections are the mechanism that enables the system to process and route documents between the Community Manager and its various participants. Connections contain the information necessary for the proper exchange of each document flow.

To create a participant connection between Partner One and Partner Two, take the following steps:

1. Click **Account Admin** from the main menu and **Participant Connections** from the horizontal navigation bar.
2. From the **Source** list, select **Partner One**.
3. From the **Target** list, select **Partner Two**.
4. Click **Search**.
5. Activate the Participant Connection that is displayed below by clicking on the **Activate** button. This should display the B2B Capabilities shown in Table 108.

Table 108. Activate Partner One-to-Partner Two participant connection

Document flow type	Source	Target
Package	None (N/A)	AS (N/A)
Protocol	EDI-X12 (ALL)	EDI-X12 (ALL)
Document Flow	ALL (ALL)	ALL (ALL)

To create a participant connection where Partner Two is the source and Partner One is the target, take the following steps:

1. Click **Account Admin** from the main menu and **Participant Connections** from the horizontal navigation bar.
2. From the **Source list**, select **Partner Two**.
3. From the **Target list**, select **Partner One**.
4. Click **Search**.
5. Activate the connection with the details shown in Table 109.

Table 109. Activate Partner Two-to-Partner One participant connection

Document flow type	Source	Target
Package	AS (N/A)	None (N/A)
Protocol	EDI-X12 (ALL)	EDI-X12 (ALL)
Document Flow	ALL (ALL)	ALL (ALL)

Creating targets

The Target List screen provides location information that enables Business Integration Connect's Document Manager to fetch documents from the appropriate system location based on the transport type of the incoming document. You can create separate target configurations based on transport type. The Document Manager can then poll the document repository locations of multiple Web, FTP, and POP mail servers--including internal directories and JMS queues--for incoming documents.

After the Document Manager retrieves a document from the location based on a pre-defined target, the routing infrastructure can process the document based on channel configuration.

To receive an EDI transaction from WebSphere Data Interchange, create a new JMS target by doing the following:

1. Click **Hub Admin** from the top-level menu.
2. Click **Targets** from the second-level menu, and then click **Create**.

3. Assign the properties shown in Table 110.

Table 110. Target properties for receipt over JMS

Target property	Value
Target Name	WdiJmsListener
Transport	JMS
Gateway Type	Production
JMS Provider URL	file:///opt/mqm/java/JNDI/WdiJms
JMS Queue Name	EDI_IN_B
JMS Factory Name	WBIC42_QM_QCF
JNDI Factory Name	com.sun.jndi.fscontext.RefFSContextFactory

A second target is required for the receipt of EDI from Partner Two via AS2. Take the following steps to create this target:

1. Click **Hub Admin** from the top level menu.
2. Click **Targets** from the second level menu, and then click **Create**.
3. Assign the properties from Table 111 below:

Table 111. Target properties for receipt over AS2

Target Name	WbicAS2Listener
Transport	HTTP/S
Gateway Type	Production
URI	/bcgreceiver/submit Note: The URI for receipt of HTTP/S must always begin with /bcgreceiver

4. Click **Save**.

Configuring Business Integration Connect - Express

This section provides you with the steps to configure the community participant's environment. In this case, this environment is handled with a WebSphere Business Integration Connect - Express system. In the sample scenario presented in this chapter, partnerTwo is using WebSphere Business Integration Connect - Express to send and receive EDI using HTTP AS2.

To successfully receive EDI via HTTP AS2, take the following steps:

1. "Configuring My Profile"
2. "Creating a participant for Partner One" on page 165
3. "Configuring the Partner One participant" on page 165

Configuring My Profile

As the first step, you must create a profile for Partner Two in WebSphere Business Integration Connect - Express. To create a profile for Partner Two, take the following steps:

1. Click on **Configuration** from the main menu.
2. Click **My Profile** from the horizontal navigation bar.

3. Enter the details as outlined in Table 112.

Table 112. My Profile details

Receipt Address Unsecure Domain	<i>IP_MACHINE_C</i> where <i>IP_MACHINE_C</i> is the Internet protocol (IP) address of Machine C, where WebSphere Business Integration Connect - Express is running
Receipt Address Unsecure Port	80 where 80 is the port assigned for use by WebSphere Business Integration Connect - Express during installation.
AS2 Sender ID	987654321
Business ID Type	DUNS
Business Identifier	987654321

4. Click **Save**.

Creating a participant for Partner One

Partner One must be identified as a participant to WebSphere Business Integration Connect - Express. To create Partner One as a participant, take the following steps:

1. Click **Configuration** from the main menu.
2. Click **Participants** from the horizontal navigation bar.
3. Click the **Create Participants** button.
4. Assign the following values:
 - a. Participant Name: **partnerOne**
 - b. AS2 Participant ID: **123456789**
5. Click **Save**.

From the Manage Participants view, you can see the details for partnerOne.

Configuring the Partner One participant

Once the participant for Partner One exists, you must configure Partner One for AS2 and HTTP. This configuration identifies the parameters required by WebSphere Business Integration Connect - Express for both sending and receiving HTTP and AS2 to partnerOne.

To configure partnerOne for HTTP and AS2, take the following steps:

1. Click **Configuration** from the main menu.
2. Click **AS2** from the horizontal navigation bar.
3. Select **partnerOne** from the **Selected Participant** list and click **Edit**.
4. Define the Outbound Destination Address of partnerOne as:
`http://IP_MACHINE_B:7080/bcgreceiver/submit`
Where *IP_MACHINE_B* is the IP address of Machine B.
5. Click **Save**.
6. Click **HTTP** from the horizontal navigation bar. (**partnerOne** should still be displayed as the Selected Participant.)
7. Click **Edit**.

8. Set the Inbound User Name and Password:
User Name: **partnerOne**
Password: **partnerOne**
Remember these were referenced earlier in the sample step of creating the default gateway for Partner Two in WebSphere Business Integration Connect Enterprise Edition on Machine B.
9. Set the Outbound Destination Address to:
`http://IP_MACHINE_B:7080/bcgreceiver/submit`
10. Click **Save**.

Important: After making these changes in WebSphere Business Integration Connect - Express, log out of the console and stop the gateway. Restart the gateway and console for all changes to take effect.

Summary

This chapter described the process by which Business Integration Connect interacts with WebSphere Data Interchange. It also provided you with procedures to set up the sample scenario described in “Sample scenario used in this chapter” on page 149.

As mentioned at the beginning of this chapter, you can follow the *Integrating WebSphere Data Interchange V3.2 with WebSphere Business Integration Connect V4.2* tutorial to actually create a sample configuration. The tutorial provides sample scripts and maps to help you configure the environment and then shows you how to test a sample exchange. To access the tutorial, go to:

www.ibm.com/developerworks/websphere/

and search on the title of the tutorial.

Chapter 7. Routing EDI documents

This section describes the process by which Business Integration Connect determines the routing information for electronic data interchange (EDI) documents it sends and receives. It describes:

- The general flow of this processing (see Overview of EDI routing)
- Additional processing required when AS packaging has been specified (see “Special considerations for AS packaging” on page 168)

You can find additional information on how file-based integration can be used when routing EDI documents in “File-system protocol for Enterprise and Advanced editions” on page 21.

Overview of EDI routing

An EDI document contains information, within the document, about the sender and the recipient of the document. Business Integration Connect uses this information when it routes the EDI document. The general flow is as follows:

1. Business Integration Connect determines the protocol used by examining the first three characters of the document. Table 113 shows the document-type protocol associated with each code.

Table 113. EDI codes and associated document types and protocols

Code	Document type	Document type protocol	Outbound as Content Type:
ISA	X12	EDI-X12	application/EDI-X12
GS	X12	EDI-X12	application/EDI-X12
UNB	Edifact	EDI-EDIFACT	application/EDIFACT
UNA	Edifact	EDI-EDIFACT	application/EDIFACT
ICS	ICS	EDI-X12	application/EDI-X12
STX	UNTDI	EDI-Consent	application/edi-consent
BG	UCS	EDI-Consent	application/edi-consent

2. Business Integration Connect extracts, from the EDI document, the sender information, based on the element and position for that particular document type, as described in Table 114.

Table 114. EDI codes and the location of the sender and receiver information

Code	From Qualifier	From ID	To Qualifier	To ID
ISA	Element 105 at position 5	Element 107 at position 6	Element 105 at position 7	Element 106 at position 8
GS	N/A	Element 142 at position 2	N/A	Element 124 at position 3
UNB UNA	Sub-element 0007 at position 2 of composite element S002 at position 20 (2nd composite) of the UNB segment	Sub-element 0004 at position 2 of composite element S002 at position 20 (2nd composite) of the UNB segment	Sub-element 0007 at position 2 of composite element S003 at position 30 (3rd composite) of the UNB segment	Sub-element 0010 at position 1 of composite element S003 at position 30 (3rd composite) of the UNB segment
ICS	Element X05 at position 4	Element X06 at position 5	Element X05 at position 6	Element X08 at position 7

Table 114. EDI codes and the location of the sender and receiver information (continued)

Code	From Qualifier	From ID	To Qualifier	To ID
STX	Element FROM1 at position 3	Element FROM2 at position 3	Element UNT1 at position 4	Element UNT2 at position 4
BG	N/A	Element BG03 at position 3	N/A	Element BG04 at position 4
UCS	N/A	Element 142 at position 3	N/A	Element 124 at position 4

- Business Integration Connect determines the sender ID from the sender ID and qualifier of the EDI document.
Note that some EDI envelopes (for example, GS) do not have the notion of qualifiers. In this case, Business Integration Connect uses only the ID.
- Business Integration Connect concatenates the qualifier and ID with a dash (-) character to look up the sender ID from the Business Integration Connect profile repository. For example, if, in the EDI message for the sender, the qualifier is AB and the identifier is 1234567, Business Integration Connect expects to find a community participant with an identifier of AB-1234567 in the profile repository. If Business Integration Connect cannot find this ID, the EDI document is not routed.
- To look up the receiving partner, Business Integration Connect determines the receiver qualifier and ID from the EDI message.
- Business Integration Connect concatenates the qualifier and ID with a dash (-) character to look up the receiver ID in the profile repository.
- Business Integration Connect routes the document to the intended recipient.

Special considerations for AS packaging

When the packaging of the document is specified as AS, Business Integration Connect performs some additional processing.

Routing the inbound document

When an EDI document is received from a community participant:

- Business Integration Connect first checks the AS1 or AS2 header information. Specifically, it checks the sender and receiver information to determine whether it matches IDs for valid community participants.
 - For AS1, it uses the Subject header field, which is in the form *"ToID;FromID"*.
 - For AS2, it uses the AS2-From and AS2-To header fields.

If the values in the header fields do not match valid IDs, Business Integration Connect does not route the document.
- Business Integration Connect then performs the steps described in "Overview of EDI routing" on page 167.

Routing the outbound document

When an EDI document is received from a back-end system, Business Integration Connect determines whether an AS BusinessID attribute has been specified for both the source packaging (None) and the target packaging (AS):

- If the AS BusinessId attribute has been specified, Business Integration Connect uses this information to generate the From and To IDs in the AS1 or AS2 header.
- If the attribute has not been specified, Business Integration Connect determines the protocol of the document, extracts the sender and receiver information and

concatenates the result (as described in “Overview of EDI routing” on page 167) and then populates the header information.

Setting both IDs in the participant profile

Because Business Integration Connect uses both the AS1 or AS2 header information as well as the information derived from the EDI document, the IDs for the same participant could be in different forms. For example, the AS header information for the sender could be 123456789 while the information derived from the EDI document could be AB-12345678.

Make sure that you have listed both IDs in the profile for the community participant. Refer to the *Administrator Guide* for information.

Part 4. Appendixes

Appendix. Configuring a JMS protocol with WebSphere MQ

This appendix describes the steps you perform to configure the JMS transport protocol in the version 5.3 IBM WebSphere MQ product. You must configure a JMS protocol because WebSphere MQ does *not* configure JMS by default. These steps use file-based JNDI to configure JMS for use with WebSphere Business Integration Connect and a back-end system.

Note: The instructions in this section assume that the WebSphere MQ queue manager is local; that is, this queue manager resides on the same machine as WebSphere Business Integration Connect. If your queue manager is remote (on some other machine), consult the WebSphere MQ documentation for information on how to set up your queues.

To configure the JMS transport protocol to send and receive documents with a back-end system, you:

1. Configure the JMS-configuration directory, which contains the files received by a JMS queue
2. Create the JMS queues and channels hosted by a queue manager.
3. Create a JMS bindings file for WebSphere MQ 5.3.
4. Create a JMS target to use the JMS inbound queue.
5. Create a JMS gateway to use the JMS outbound queue.

Configuring the JMS-configuration directory

The JMS-configuration directory will contain the files that come into the server on a JMS queue that is hosted by a WebSphere MQ queue manager. To create the JMS bindings file, you use the JMSAdmin application. However, before you run this application, you must customize its configuration file to your JMS environment. configure the JMS

Note: The WebSphere MQ documentation provides a full description on how to create a JMS bindings file. This section provides an overview of that process.

To configure the JMS-configuration directory, take the following steps:

1. Create a JMS-configuration directory, somewhere on the machine on which Business Integration Connect resides.
2. Open the JMSAdmin application's configuration file, `JMSAdmin.config`, for edit. You must customize this file to provide the JMSAdmin application with information about your JMS configuration. This file resides in the following directory:

```
WebSphereMQ_Root\java\bin
```

where *WebSphereMQ_Root* is the product directory of WebSphere MQ.

3. Comment out the following lines by inserting a pound character (#) as the first character of the line:

```
INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory  
PROVIDER_URL=ldap://polaris/o=ibm,c=us
```

4. Remove the comment from the following lines by deleting the initial pound character (#) as the first character of the line:

```
#INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.ReffSContextFactory
#PROVIDER_URL=file://C:/JNDI-Directory
```

5. Change the path for the PROVIDER_URL variable to the JMS-configuration directory. This is the directory you created in step 1 above.

Note: This directory must exist and your user account must have write permission to this folder.

6. Save the JMSAdmin.config file.

Suppose you create your JMS-configuration directory as follows:

```
C:/filesender/config
```

For this JMS-configuration directory, Figure 29 shows the modified lines of the JMSAdmin.config file.

```
#INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.ReffSContextFactory
#
# The following line specifies the URL of the service provider's initial
# context. It currently refers to an LDAP root center. Examples of a
# file system URL and WebSphere's JNDI namespace are also shown, commented
# out
#PROVIDER_URL=ldap://polaris/o=ibm,c=us
PROVIDER_URL=file://C:/filesender/config
```

Figure 29. Sample lines of the JMSAdmin.config file

Creating the JMS queues

The JMS transport protocol uses JMS queues to hold the JMS messages passed between Business Integration Connect and the back-end system. With WebSphere MQ version 5.3, creation of the JMS queues involves the following steps:

1. "Creating the MQ queue manager"
2. "Creating the MQ channels and transmission queue" on page 175
3. "Creating the MQ JMS local queues" on page 176

Creating the MQ queue manager

If you have not already defined a queue manager for Business Integration Connect and the back-end system, you must do so before creating the MQ JMS queues. You will need a queue manager if you are doing JMS integration. You must decide whether to install a new queue manager or to use an existing one. An existing queue manager can be the same queue manager as Business Integration Connect, the same queue manager as the back-end system, or another existing queue manager. Refer to the WebSphere MQ documentation for instructions on how to create a queue manager.

Note: If your back-end system and Business Integration Connect reside on different machines, the queue manager can reside either machine. Both Business Integration Connect and the back-end system specify the machine host name when they access the queue manager.

The samples in this chapter assume that a WebSphere MQ queue manager exists and is named as follows:

```
bcg.queue.manager
```

Creating the MQ channels and transmission queue

Once the queue manager exists, make sure that the following objects are created:

- Transmission queue
- Remote queue
- Receiver queue
- Sender channel
- Receiver channel

To create the MQ channels and transmission queue, use the WebSphere MQ Command Environment, `runmqsc`, which WebSphere MQ provides. This tool provides a command-line interface for you to enter queue-management commands.

Note: The way you create these MQ objects depends on the platform you are using. Refer to the WebSphere MQ documentation for instructions on creating these objects.

To use `runmqsc` tool to create your MQ channels and transmission queue, follow these steps:

1. Open a command prompt and move to the following directory:

```
WebSphereMQ_Root\java\bin
```

where *WebSphereMQ_Root* is the installation directory of WebSphere MQ.

2. Enter the following command to start the WebSphere MQ Command Environment:

```
runmqsc queueManager
```

where *queueManager* is the name of your WebSphere MQ queue manager.

Note: You *must* be logged in as the `mqm` user to use the `runmqsc` tool.

3. At the `runmqsc` command line, you can enter the following commands:

- a. Define a JMS transmission queue:

```
define qlocal(transQueueName) usage(xmitq) put(enabled)  
get(enabled)
```

where *transQueueName* is the desired name of your JMS transmission queue.

- b. Define a sender channel:

```
define channel(sndrChannelName) chltype(sdr) xmitq(transQueueName)  
conname('remote m/c ip')  
stop channel(sndrChannelName)
```

where *sndrChannelName* is the desired name of your JMS sender channel and *transQueueName* is the name of the JMS transmission queue you created in step 3a.

- c. Define a receiver channel:

```
define channel(rcvrChannelName) chltype(rcvr) xmitq(transQueueName)  
stop channel(rcvrChannelName)
```

where *rcvrChannelName* is the desired name of your JMS receiver channel and *transQueueName* is the name of the JMS transmission queue you created in step 3a.

4. You can leave your WebSphere MQ Command Environment open, as you will need it for subsequent queue-management commands.

Figure 30 shows the creation of sample JMS channels and a transmission queue hosted by the `bcg.queue.manager` queue manager.

```

runmqsc bcg.queue.manager
  define qlocal(TRANSQ) usage(xmitq) put(enabled) get(enabled)
  define channel(SENDER) chltype(sdr) xmitq(TRANSQ)
    conname('remote m/c ip')
  stop channel(SENDER)
  define channel(RECEIVER) chltype(rcvr)
  stop channel(RECEIVER)

```

Figure 30. Commands to create sample JMS channels and transmission queue

The commands in Figure 30 create the following MQ objects:

- A transmission queue named TRANSQ
- A sender channel named SENDER
- A receiver channel named RECEIVER

Creating the MQ JMS local queues

To create the MQ JMS local queues, use the WebSphere MQ Command Environment, runmqsc.

Note: You must create these MQ JMS queues on the machine on which Business Integration Connect resides.

To use the runmqsc tool to create your local MQ queues, follow these steps:

1. At the runmqsc command line, you can enter the following commands:
 - a. Define the JMS inbound queue, which receives messages from the back-end system:


```
define qlocal(inQueueName)
```

 where *inQueueName* is the desired name of your JMS inbound queue.
 - b. Define the JMS outbound queue, which sends messages to the back-end system:


```
define qlocal(outQueueName)
```

 where *outQueueName* is the desired name of your JMS outbound queue.
2. Exit the WebSphere MQ Command Environment:


```
end
```

You can leave your command prompt open, as you will need it in subsequent configuration steps.

Note: If your interactions with Business Integration Connect involve only *one* direction of communication with the back-end system, you can create only the queue for the direction Business Integration Connect will support.

Figure 31 shows the creation of sample JMS queues hosted by the bcg.queue.manager queue manager.

```

define qlocal(JMSIN)
define qlocal(JMSOUT)

```

Figure 31. Commands to create sample JMS queues

The commands in Figure 31 create the following JMS queues:

- An local inbound queue named JMSIN
- An local outbound queue named JMSOUT

Creating the JMS bindings file

To create the JMS bindings file, you use the JMSAdmin application. This section provides a summary of how to create the JMS bindings file. For complete information on how to use the JMSAdmin application, refer to your WebSphere MQ documentation.

The following steps describe how to create the JMS bindings file:

1. In a command prompt (from the *WebSphereMQ_Root\java\bin* directory), launch the JMSAdmin application by entering:
JMSAdmin
2. Define the new JMS configuration by entering the following commands at the command-line prompt:
 - a. Create the JMS context with the following command:
`def ctx(contextName)`
 - b. Change your active context by typing:
`chg ctx(contextName)`
where *contextName* is the context you created in the previous step.
 - c. Define the queue connection factory with the following command:
`def qcf(connectionFactoryName) qmgr (queueManagerName) tran(client)
chan(javaChannelName) host (MQHostName) port (MQport)`
where:
 - *connectionFactoryName* is the name to assign to your queue connection factory
 - *queueManagerName* is the name of your WebSphere MQ queue manager (that manages the queues that Business Integration Connect will use)
 - *javaChannelName* is the name of the channel used to establish client communications to WebSphere MQ. The default channel name is `java.channel`.
 - *MQHostName* is the IP address of the host machine (the machine on which the WebSphere MQ queue manager resides)
 - *MQport* is the port number of the host machine
 - d. Define the queues by typing the following command *for each queue*:
`def q(queueAliasName) qmgr (queueManagerName) queue (queueName)`
 - e. Exit JMSAdmin with the following command:
end

The bindings file is created in a subfolder of the folder configured in the PROVIDER_URL field of the JMSAdmin.config file. The bindings file has the following name:

```
.bindings
```

The name of the subfolder is the name you chose for your JMS context.

Figure 32 shows the creation of a sample JMS configuration.

```

def ctx(JMS)
change ctx(JMS)
define qcf(WBICHub) qmgr(bcg.queue.manager) tran(CLIENT) chan(java.channel)
host(127.0.0.1) port(1414)
define q(inQ) queue (JMSIN) qmgr(bcg.queue.manager)
define q(outQ) queue (JMSOUT) qmgr(bcg.queue.manager)

```

Figure 32. Commands to create sample JMS configuration

The commands in Figure 32, issued from within the JMSAdmin application, create the following JMS objects:

- A JMS context named JMS
- A queue alias, inQ, for the local inbound queue (JMSIN)
- A queue alias, outQ, for the local outbound queue (JMSOUT)
- A queue connection factory named WBICHub

When these commands are complete, the JMSAdmin application has created a .bindings file in the following directory:

```
C:/filesender/config/jms
```

Creating the JMS target

Copy the bindings file created in “Configuring the JMS-configuration directory” on page 173 to the directory where you want it to reside. If you want to preserve the JMS context, copy the subfolder (with the same name as the context) and the bindings file to the directory, so that the full path to the bindings file is as follows:

```
/parentDirectory/contextSubdirectory/.bindings
```

A sample path for the bindings file is:

```
/mydir/myctx/.bindings
```

From the Targets screen of the Community Console, create a target, specifying the following information shown in Table 115

Table 115. Information for Target Details screen of the Community Console

Field name	Value	Example
Transport	JMS	<i>Same</i>
JMS Provider URL	The file-system path to the JMS-configuration directory, where the context subfolder (if there is a context) and the bindings file are located, in the following form: <i>file://JMSConfigDirectory</i>	This value includes the JMS context in the JMS Provider URL: <i>file:///C:/filesender/config/jms</i>

where *JMSConfigDirectory* is the full pathname.
Optionally, this *JMSConfigDirectory* can contain the context subdirectory, where the .bindings file resides.

Table 115. Information for Target Details screen of the Community Console (continued)

Field name	Value	Example
JMS Queue Name	<p>The JMS queue alias name that you specified when creating the JMS bindings file. This queue alias name is relative to the path you specified in the JMS Provider URL field:</p> <ul style="list-style-type: none"> • If the JMS Provider URL includes the context name, you do not need to provide the context name as part of the JMS queue alias name. • If the JMS Provider URL does <i>not</i> include the context name, you must provide it in the JMS queue alias name, in the form: <i>JMScontext/JMSqueueAlias</i> 	inQ
JMS Factory Name	<p>The queue connection factory. You specified this name with the <code>define qcf</code> command when creating the bindings file. This factory name is relative to the path you specified in the JMS Provider URL field:</p> <ul style="list-style-type: none"> • If the JMS Provider URL includes the context name, you do not need to provide the context name as part of the JMS factory name. • If the JMS Provider URL does <i>not</i> include the context name, you must provide it in the JMS factory name, in the form: <i>JMScontext/JMSfactory</i> 	WBICHub
JNDI Factory Name	com.sun.jndi.fscontext.RefFSContextFactory	Same

Note: The target must be able to access the directory where the subfolder and JMS bindings file are located.

In Table 115, the Example column lists the target values for the JMS configuration defined by Figure 31 on page 176 and Figure 32 on page 178.

Creating the JMS gateway

From the Gateways screen of the Community Console, create a gateway, specifying the information in Table 116

Table 116. Information for Gateway Details screen of the Community Console

Field name	Value	Example
Transport	JMS	Same
Target URI	<p>The file-system path to the JMS-configuration directory, where the context subfolder (if there is a context) and the bindings file are located, in the following form: <i>file://JMSConfigDirectory</i></p> <p>where <i>JMSConfigDirectory</i> is the full pathname. Optionally, this <i>JMSConfigDirectory</i> can contain the context subdirectory, where the <code>.bindings</code> file resides.</p>	<p>This value includes the JMS context in the Target URI: <code>file://C:/filesender/config/jms</code></p>

Table 116. Information for Gateway Details screen of the Community Console (continued)

Field name	Value	Example
JMS Factory Name	<p>The JMS factory name that you specified when creating the JMS bindings file. This factory name is relative to the path you specified in the Target URI field:</p> <ul style="list-style-type: none"> • If the Target URI includes the context name, you do not need to provide the context name as part of the JMS factory name. • If the Target URI does <i>not</i> include the context name, you must provide it in the JMS factory name, in the form: <i>JMScontext/JMSfactory</i> 	WBICHub
JMS Message Class	<p>One of the following JMS message classes:</p> <ul style="list-style-type: none"> • StreamMessage • BytesMessage • TextMessage 	<i>Depends on message class supported by back-end system</i>
JMS Queue Name	<p>The JMS queue alias name that you specified when creating the JMS bindings file. This queue alias name is relative to the path you specified in the Target URI field:</p> <ul style="list-style-type: none"> • If the Target URI includes the context name, you do not need to provide the context name as part of the JMS queue alias name. • If the Target URI does <i>not</i> include the context name, you must provide it in the JMS queue alias name, in the form: <i>JMScontext/JMSqueueAlias</i> 	outQ
JNDI Factory Name	com.sun.jndi.fscontext.ReffSContextFactory	<i>Same</i>

In Table 116 on page 179, the Example column lists the gateway values for the JMS configuration defined by Figure 31 on page 176 and Figure 32 on page 178.

Notices and Trademarks

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licenses of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Burlingame Laboratory Director

IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not necessarily tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Websphere Business Integration Connect contains code named ICU4J which is licensed to you by IBM under the terms of the International Program License Agreement, subject to its Excluded Components terms. However, IBM is required to provide the following language to you as a notice:

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2003 International Business Machines Corporation and others

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM
the IBM logo
AIX
CrossWorlds
DB2
DB2 Universal Database
Domino
Lotus
Lotus Notes
MQIntegrator

MQSeries
Tivoli
WebSphere

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

WebSphere Business Integration Connect Enterprise and Advanced Editions includes software developed by the Eclipse Project (www.eclipse.org).



WebSphere Business Integration Connect Enterprise and Advanced Editions
Version 4.2.2

Index

A

- Access client 59, 69
- Adapter for HTTP 96
 - binding to collaboration 110
 - business-object structure 102
 - configuring 100
 - installing 100
 - payload data handler 100
 - protocol handler 100
- Adapter for JMS
 - binding to collaboration 126
 - business-object structure 121
 - configuring 120
 - installing 120
 - payload data handler 120
 - setting the input queue 121
- Adapter for Web Services 111
- Adapter for XML
 - binding to collaboration 95
 - business-object structure 88
 - configuring 85, 86
 - installing 85
 - payload data handler 86
- AS packaging 168
- AS1 document 15
- AS2 document 13, 16
- Asynchronous interaction
 - cXML documents 6
 - HTTP transport protocol 19
 - ICS over HTTP 70, 71, 92, 106, 122
 - ICS over JMS 40
 - JMS transport protocol 20
 - Message Broker over HTTP 130
 - Message Broker over JMS 131
- Attachment 11, 17
 - attachment container 64, 65
 - attachment data 62
 - content information 62
 - content type 17
 - data handler for 49
 - encoding 17
- Attachment business object 63
- Attachment data handler 49, 68, 96
 - configuring 56
 - creating business object definitions 60
 - creating configuration business objects 56
 - location of 55
 - repository file 55, 58
 - representing attachments 63
 - SOAP documents 111
- Attachment-container business object 64

B

- B2B capabilities 28, 33, 159
- Back-end document 31
- Back-end integration 3
 - overview 3
- Back-end system 3
 - choosing 22

- Back-end system (*continued*)
 - receiving documents from 31
 - sending documents to 26
- Backend Integration packaging 11
 - business objects for HTTP 87, 102
 - business objects for JMS 122
 - Envelope Flag 16
 - example 18
 - HTTP header information 92, 106
 - HTTP transport protocol 19
 - JMS transport protocol 20
 - JMS transport-level header 122
 - when required 17, 19, 21
- Binary document 27, 32
- Business Integration Connect
 - business protocols 5
 - configuring 25
 - configuring for Data Interchange 157
 - configuring for InterChange Server 41
 - configuring for Message Broker 131
 - packaging types 10
 - receiving documents 31
 - from Data Interchange 163
 - from ICS 42
 - from Message Broker 133
 - sending documents 26
 - to Data Interchange 160
 - to ICS 41
 - to Message Broker 131
- Business object 44
 - attachment-container 64
 - content-information 62
 - default attachment 62
 - dynamic meta-object 94, 108, 124
 - event 47
 - for document 45
 - HTTP protocol-configuration meta-object 108
 - HTTP-properties 93
 - JMS-properties 123
 - payload 45, 65
 - request.
 - See* Request business object response.
 - See* Response business object top-level 89, 103
 - user-defined-properties 107
- Business object definition 44
 - Attachment data handler 60
 - creating 44, 60, 87, 102, 121
 - for HTTP 87, 102
 - for JMS 121
- Business protocol 5

C

- Collaboration 48
 - Adapter for HTTP 110
 - Adapter for JMS 126
 - Adapter for XML 95
 - Connect Servlet 74
- Community Manager 3

Compute node 137, 139, 144
Connect Servlet 39, 68, 69
 configuring 71
 deploying 72
 identifying collaborations for 74
 location of 72
 servlet properties file 73
Content-information business object 62
cXML document 4, 6, 19, 92

D

Data handler 39
 Attachment 49
 child meta-object 56, 77
 top-level meta-object 58, 79
 Wrapper 76
Data Interchange 147
Default attachment business object 62
Document
 AS1 15
 AS2 13, 16
 attachments.
 See Attachment
 back-end 31
 business object for 45
 cXML 6, 92
 EDI 147, 167
 participant 31
 payload.
 See Payload
 RosettaNet 7, 12
 SOAP 6, 110, 138
 transport envelope 16, 17, 45
Document flow definition 28, 162
Document flow interaction 33, 162
Dynamic meta-object
 for HTTP 92, 94, 106, 108
 for JMS 124

E

EDI document 4, 147
 HTTP transport protocol 19
 location of payload 16
 packaging for 27, 32
 routing 167
 supported transport protocols 27, 32
Event business object 47
Event notification 7, 47
 over HTTP 68, 70, 80, 85, 100
 over JMS 120

F

Fault business object 104
File-system transport protocol 21

G

Gateway 27
 HTTP transport protocol 41, 132
 JMS transport protocol 42, 132, 179
 to Data Interchange 160
 to InterChange Server 41

Gateway (*continued*)
 to Message Broker 131

H

HTTP protocol handler 84, 85, 86, 100
HTTP protocol-configuration meta-object 108
HTTP transport protocol 19
 Business Integration Connect
 and ICS 41, 43
 and Message Broker 132, 133
 creating data business object 122
 creating header information 92, 106
 ICS business-object structure 87, 102
 ICS integration samples 40
 InterChange Server and 39, 67, 95, 110
 location of payload 16
 Message Broker and 130, 135
 receiving documents
 from ICS 81, 98
 from Message Broker 136
 sending documents
 to ICS 67, 97
 to Message Broker 135
 transport-protocol mechanism 25
HTTP-properties business object 93
HTTPInput node 137, 138
HTTPReply node 137, 139
HTTPRequest node 137, 139

I

ICS.
 See InterChange Server
InputDestination connector-specific property 121
Interaction.
 See Asynchronous interaction, Synchronous interaction
InterChange Server 37
 configuring 44, 67, 95, 109, 125
 creating artifacts for 95, 109, 125
 event notification 70, 80, 85, 100, 120
 HTTP transport protocol 67, 95, 110
 integrating with 113
 integration samples 40
 InterChange Server-compatible component 38
 JMS transport protocol 113
 planning integration for 37
 request processing 85, 87, 100, 120
 required components for receiving
 over HTTP 81, 96
 over JMS 113
 required components for sending
 over HTTP 68, 96
 over JMS 113
 supported transport protocols 38
 supported versions 38

J

JMS bindings file 173
JMS transport protocol 20
 Business Integration Connect
 and Data Interchange 160, 163
 and ICS 42, 43
 and Message Broker 132, 134
 creating header information 122

- JMS transport protocol (*continued*)
 - ICS business-object structure 121
 - ICS integration samples 40
 - InterChange Server and 40, 113
 - location of payload 16
 - Message Broker and 130, 131, 139
 - receiving documents
 - from ICS 116
 - from Message Broker 141
 - sending documents
 - to ICS 114
 - to Message Broker 139
 - transport-protocol mechanism 25
 - WebSphere MQ 173
- JMS-properties business object 123

M

- Message Broker 129
 - configuring 134
 - HTTP transport protocol 135
 - integration samples 131
 - JMS transport protocol 139
 - planning integration for 130
 - required components for receiving
 - over HTTP 135
 - over JMS 139
 - required components for sending
 - over HTTP 135
 - over JMS 139
 - supported transport protocols 130
 - supported versions 130
- Message flow 134, 136, 138, 144
- Message.
 - See* Document
- MO_DataHandler_Default top-level meta-object 58
- MO_Server_DataHandler top-level meta-object 58, 77, 79
- MQInput node 137, 144
- MQOutput node 137, 144

N

- None packaging 11
 - business objects for HTTP 87, 102
 - business objects for JMS 122
 - file-system protocol 21
 - HTTP transport protocol 19, 21
 - when required 17, 19, 21

P

- Packaging 10
 - Backend Integration 11
 - None 11
- Participant connection 30, 33, 162
- Participant document 31
- Payload 11, 16, 45
 - content type 17
 - encoding 17
- Payload business object 45, 65
 - for HTTP 88, 102
 - for JMS 122
- Payload data handler 46
 - Adapter for HTTP 100
 - Adapter for JMS 120
 - Adapter for XML 86

R

- Request business object 47
 - incorporating message-header information 94, 108
 - location of 90, 104
 - role of 80, 87
 - structure of 90, 104
- Request processing 47
 - over HTTP 81, 85, 87, 100
 - over JMS 120
- Response business object 47, 105
 - location of 90, 104
 - role of 81, 87
 - structure of 91
- RosettaNet document 7, 12
 - ICS integration samples 40
 - location of payload 16
 - packaging for 19, 27, 32
 - supported transport protocols 27, 32

S

- SOAP document 4, 6
 - HTTP transport protocol 19
 - ICS over HTTP 110
 - Message Broker over HTTP 138
 - packaging for 27, 32
 - supported transport protocols 27, 32
- Synchronous interaction
 - HTTP transport protocol 19
 - ICS over HTTP 40, 70, 71, 86, 91, 105
 - Message Broker over HTTP 130

T

- Target 31
 - from Data Interchange 163
 - from InterChange Server 43
 - from Message Broker 133
 - HTTP transport protocol 43, 133
 - JMS transport protocol 43, 134, 178
- Top-level business object 89, 103
- Transport envelope 16, 17, 45, 49
- Transport protocol 18
 - choosing 38, 130
 - location of payload 16
- Transport-level header 11
 - for HTTP 92, 106
 - for JMS 122
- Transport-protocol mechanism 24
 - HTTP 25
 - JMS 25

U

- User-defined-properties business object 107

W

- WebSphere Business Integration Adapter for HTTP.
 - See* Adapter for HTTP
- WebSphere Business Integration Adapter for JMS.
 - See* Adapter for JMS
- WebSphere Business Integration Adapter for Web Services.
 - See* Adapter for Web Services

WebSphere Business Integration Adapter for XML.
 See Adapter for XML

WebSphere Business Integration Connect Servlet.
 See Business Integration Connect Servlet

WebSphere Business Integration Connect.
 See Business Integration Connect

WebSphere Business Integration Message Broker.
 See Message Broker

WebSphere Data Interchange.
 See Data Interchange

WebSphere InterChange Server.
 See InterChange Server

WebSphere MQ 173

Wrapper data handler 68, 76
 business-object structure 88
 creating configuration business objects 77
 location of 72, 77
 repository file 72, 78

X

XML transport envelope.
 See Transport envelope



Printed in USA