B2B Solutions using WebSphere BI Connect
Version 4.2.2

Implement EDI solutions using WebSphere BI Connect, WebSphere Data Interchange

Extend the features of WebSphere BI Connect

Enable RosettaNet support in WebSphere BI Connect

Geert Van de Putte
Jeffrey Brent
Ronan Dalton
Rich Kinard
Benjamin Thurgood
B2B Solutions using WebSphere BI Connect
Version 4.2.2

March 2005
Note: Before using this information and the product it supports, read the information in “Notices” on page xxxi. Published in co-operation with RosettaNet.


This edition applies to Version 4, Release 2, Modification 2 of WebSphere BI Connect Advanced (product number 5724-E75) and Version 4, Release 2, Modification 2 of WebSphere BI Connect Enterprise (product number 5724-E78).

© Copyright International Business Machines Corporation 2004, 2005. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures ......................................................................................... xiii

Tables ......................................................................................... xxvii

Examples ..................................................................................... xxix

Notices ........................................................................................ xxxi

Trademarks .................................................................................. xxxii

Preface ......................................................................................... xxxiii

The team that wrote this redbook ........................................... xxxiii

Become a published author ...................................................... xxxv

Comments welcome .................................................................... xxxv

Part 1. Overview of B2B technology and WebSphere BI Connect ........................................... 1

Chapter 1. Business-to-business concepts ........................................... 3

1.1 Impact of the Internet on the world of business applications ................................. 4

1.2 E-commerce ............................................................................... 4

1.2.1 Business-to-consumer .......................................................... 5

1.2.2 Business-to-business .......................................................... 5

1.2.3 Evolution of the B2B data structures .................................... 7

1.2.4 Evolution of B2B data communications ................................ 9

1.3 Enterprise application integration and B2B ........................................... 10

1.4 B2B integration ......................................................................... 12

1.4.1 Types of B2B integration .................................................... 13

1.4.2 Summary ............................................................................ 19

Chapter 2. B2B technologies and standards ........................................... 21

2.1 Requirements for a B2B solution ................................................ 22

2.2 Some terminology ..................................................................... 23

2.2.1 Messaging and queuing ....................................................... 23

2.2.2 Electronic data interchange ................................................ 25

2.2.3 Transport protocols ............................................................ 26

2.2.4 Security ............................................................................ 26

2.2.5 Java ..................................................................................... 29

2.2.6 Extensible Markup Language .......................................... 29

2.2.7 Web services ................................................................. 29

Chapter 3. Introduction to WebSphere BI Connect ........................................... 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Features of WebSphere BI Connect</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>Architecture of WebSphere BI Connect</td>
<td>32</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Run-time components</td>
<td>33</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Configuration components</td>
<td>36</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Profile management</td>
<td>38</td>
</tr>
<tr>
<td>3.3</td>
<td>B2B features and standards</td>
<td>39</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Transport options</td>
<td>39</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Messaging protocol options</td>
<td>40</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Business document formats</td>
<td>40</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Security options in WebSphere BI Connect</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Integration options</td>
<td>41</td>
</tr>
<tr>
<td>Part 2</td>
<td>Building a B2B exchange</td>
<td>43</td>
</tr>
</tbody>
</table>

**Chapter 4. Implementation scenarios**

4.1 Implementation of WebSphere BI Connect Enterprise for Windows       | 46   |
4.2 Implementation of WebSphere BI Connect Advanced for AIX             | 47   |
4.3 Implementing a basic exchange                                       | 47   |
4.4 Securing the exchange between Companies E and A                     | 49   |
4.5 Implementing WebSphere BI Connect Express for Windows               | 50   |
4.6 Using FTP with WebSphere BI Connect                                 | 51   |
4.7 Managing a B2B infrastructure                                        | 52   |
4.8 EDI translation                                                     | 53   |
4.9 Extending WebSphere BI Connect                                      | 55   |
4.10 Implementing RosettaNet                                            | 56   |

**Chapter 5. Implementing WebSphere BI Connect Enterprise in a Windows environment**

5.1 Implementation overview                                            | 58   |
5.2 Verifying software levels on the hub and data machine               | 58   |
| 5.2.1 Verifying WebSphere MQ                                          | 59   |
| 5.2.2 Verifying DB2                                                    | 61   |
5.3 Enabling DB2 stored procedures on the data machine                  | 62   |
| 5.3.1 Installing .NET Framework and SDK                              | 62   |
| 5.3.2 Configuring DB2 to use the .NET SDK                             | 64   |
5.4 Installing the software for the data machine                        | 65   |
| 5.4.1 Adding user IDs and a group                                      | 66   |
| 5.4.2 Configuring WebSphere MQ                                        | 68   |
| 5.4.3 Installing the database schema                                   | 73   |
| 5.4.4 Local validation                                                | 79   |
5.5 Installing the software for the hub machine                         | 80   |
| 5.5.1 Adding a user and group                                          | 80   |
| 5.5.2 Installing the product code                                     | 81   |
| 5.5.3 Local validation                                                | 87   |
Contents

5.6 Initial configuration of the WebSphere BI Connect server................. 89

Chapter 6. Implementing WebSphere BI Connect Advanced for AIX .... 95
6.1 Implementation overview........................................... 96
6.2 Verifying software levels on the AIX machine .......................... 96
  6.2.1 Verifying DB2 .................................................. 96
  6.2.2 Verifying WebSphere MQ ................................. 98
6.3 Enabling DB2 stored procedures .................................. 100
6.4 Software installation ........................................... 101
  6.4.1 Adding user accounts ...................................... 102
  6.4.2 Configuring WebSphere MQ ................................ 102
  6.4.3 Installing the database schema.................................. 105
  6.4.4 Installing the product code .................................. 113
  6.4.5 Local validation ........................................... 120
6.5 Initial configuration of the WebSphere BI Connect server ............ 120

Chapter 7. Creating a basic B2B exchange ................................. 121
7.1 Scenario overview .................................................. 122
  7.1.1 Role-based configuration .................................... 122
  7.1.2 Outbound flow .................................................. 123
  7.1.3 Inbound flow .................................................. 125
7.2 Configurations tasks for hubadmin of Company E ..................... 126
  7.2.1 Creating targets .............................................. 127
  7.2.2 Creating interactions ....................................... 132
  7.2.3 Creating a community manager ............................ 139
  7.2.4 Creating a community participant ........................ 145
7.3 Configuration tasks by Company E administrator ....................... 148
  7.3.1 Initial logon by community manager ...................... 148
  7.3.2 Creating a gateway ........................................... 149
  7.3.3 Providing B2B capabilities ................................ 154
7.4 Configuration tasks for partner Company A .......................... 156
7.5 Connecting Company E to Company A ................................ 163
7.6 Configuration tasks for hubadmin of Company A ..................... 170
7.7 Validating communication ........................................ 178
7.8 Revisiting role-based configuration ................................ 184

Chapter 8. Securing the B2B exchange ........................................ 189
8.1 What is needed to perform encryption and decryption ............... 190
8.2 Enabling encryption .............................................. 194
  8.2.1 Company E generates a public/private key pair ............ 194
  8.2.2 Company E uploads private key to its own server .......... 199
  8.2.3 Company E uploads public certificate to partner’s server .. 204
  8.2.4 What happens next? ........................................... 207
  8.2.5 Company A generates a public/private key pair............ 207
11.4 Implementing FTPs ............................... 319
  11.4.1 Enabling FTPS for the FTP server of Company F .......................... 320
  11.4.2 Uploading certificates in WebSphere BI Connect of Company E .......................... 329
  11.4.3 Updating the FTP gateway ........................................ 332
  11.4.4 Validating outbound communication .................................. 334
  11.4.5 Using FTPS for inbound communication. .............................. 334

11.5 Summary ........................................ 335

Chapter 12. Managing the B2B exchange ......................... 337
  12.1 Overview of tools to manage the exchange ......................... 338
  12.2 System log files ........................................ 339
  12.3 Event Viewer ............................................ 339
  12.4 Document Viewer ........................................ 345
  12.5 AS1/AS2 Viewer ........................................ 347
  12.6 Gateway queue ........................................... 350
  12.7 Using the tools to solve problems ............................... 354
    12.7.1 MDN HTTP URL not defined. ................................ 354
    12.7.2 Problems with encryption .................................. 356
    12.7.3 Problems with digital signatures ............................ 358

Part 3. Integration with WebSphere Data Interchange ............. 365

Chapter 13. Introduction to EDI technology and WebSphere Data Interchange ................. 367
  13.1 EDI terms and concepts ...................................... 368
  13.2 Benefits of EDI ........................................... 370
  13.3 EDI components ............................................ 371
    13.3.1 Message standards ...................................... 371
    13.3.2 Communication ......................................... 373
  13.4 The evolution of EDI ........................................ 375
    13.4.1 Elements of an EDI solution ................................ 377
    13.4.2 The IBM EDI solution .................................... 378
    13.4.3 Features of WebSphere Data Interchange .................... 379
  13.5 Usage patterns for WebSphere Data Interchange ................. 381
    13.5.1 A point-to-point solution ................................ 381
    13.5.2 An integration broker solution .............................. 382
    13.5.3 A B2B gateway solution .................................... 383
  13.6 Internet references ........................................ 384

Chapter 14. WebSphere Data Interchange infrastructure and implementation ................. 385
  14.1 The system view of WebSphere Data Interchange .................. 386
    14.1.1 Development environment ................................... 386
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1.2</td>
<td>Run-time environment</td>
</tr>
<tr>
<td>14.2</td>
<td>Step-by-step implementation</td>
</tr>
<tr>
<td>14.2.1</td>
<td>WebSphere Data Interchange Server installation and setup</td>
</tr>
<tr>
<td>14.2.2</td>
<td>Setting up the WebSphere Data Interchange database</td>
</tr>
<tr>
<td>14.2.3</td>
<td>Setting up WebSphere MQ objects</td>
</tr>
<tr>
<td>14.2.4</td>
<td>WebSphere Data Interchange Client installation and setup</td>
</tr>
<tr>
<td>14.2.5</td>
<td>Setting up a connection to the database</td>
</tr>
<tr>
<td>14.3</td>
<td>The component view of WebSphere Data Interchange</td>
</tr>
<tr>
<td>14.3.1</td>
<td>Mailbox profiles</td>
</tr>
<tr>
<td>14.3.2</td>
<td>Network profiles</td>
</tr>
<tr>
<td>14.3.3</td>
<td>WebSphere MQ-related artifacts</td>
</tr>
<tr>
<td>14.3.4</td>
<td>Service profiles</td>
</tr>
<tr>
<td>14.3.5</td>
<td>Trading Partner profiles</td>
</tr>
<tr>
<td>14.3.6</td>
<td>Rules</td>
</tr>
<tr>
<td>14.3.7</td>
<td>Final view of a running WebSphere Data Interchange system</td>
</tr>
<tr>
<td>15.1</td>
<td>Overview of implemented scenarios</td>
</tr>
<tr>
<td>15.1.1</td>
<td>Phase 1: Transforming XML into EDI for Company A</td>
</tr>
<tr>
<td>15.1.2</td>
<td>Phase 2: EDI for Company A and XML for Company X</td>
</tr>
<tr>
<td>15.1.3</td>
<td>Phase 3: Managing business identifiers</td>
</tr>
<tr>
<td>15.1.4</td>
<td>Phase 4: Multiple EDI transactions and internal applications</td>
</tr>
<tr>
<td>15.2</td>
<td>Setting up JMS for WebSphere BI Connect</td>
</tr>
<tr>
<td>15.2.1</td>
<td>Connecting the queue managers</td>
</tr>
<tr>
<td>15.2.2</td>
<td>Enabling JMS</td>
</tr>
<tr>
<td>15.2.3</td>
<td>Creating the JMS gateway</td>
</tr>
<tr>
<td>15.2.4</td>
<td>Creating the JMS target</td>
</tr>
<tr>
<td>15.2.5</td>
<td>Validating the JMS and MQ configuration</td>
</tr>
<tr>
<td>15.3</td>
<td>Handling the inbound document flow</td>
</tr>
<tr>
<td>15.3.1</td>
<td>Document definition for XML</td>
</tr>
<tr>
<td>15.3.2</td>
<td>Document definition for EDI</td>
</tr>
<tr>
<td>15.3.3</td>
<td>Transformation map for EDI to XML</td>
</tr>
<tr>
<td>15.3.4</td>
<td>Trading partner setup</td>
</tr>
<tr>
<td>15.3.5</td>
<td>Rule or usage for the transformation map</td>
</tr>
<tr>
<td>15.3.6</td>
<td>Supporting definitions</td>
</tr>
<tr>
<td>15.3.7</td>
<td>Putting it all together</td>
</tr>
<tr>
<td>15.3.8</td>
<td>Validating the map and the supporting setup</td>
</tr>
<tr>
<td>15.4</td>
<td>Handling the outbound document flow</td>
</tr>
<tr>
<td>15.4.1</td>
<td>Detailed review of the map XML-XML-850</td>
</tr>
<tr>
<td>15.4.2</td>
<td>Detailed review of the map XML-EDI-850</td>
</tr>
<tr>
<td>15.4.3</td>
<td>Detailed review of the map XML-ROUTER</td>
</tr>
<tr>
<td>15.4.4</td>
<td>Additional supporting definitions</td>
</tr>
<tr>
<td>15.5</td>
<td>Using a translation table</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>15.5.1 Validating the maps and supporting definitions</td>
<td>481</td>
</tr>
<tr>
<td>15.6 Adding a new EDI transaction</td>
<td>484</td>
</tr>
<tr>
<td>15.7 Summary</td>
<td>486</td>
</tr>
<tr>
<td>Part 4. Extending the features of WebSphere BI Connect</td>
<td></td>
</tr>
<tr>
<td>Chapter 16. Introduction to the user exit framework</td>
<td></td>
</tr>
<tr>
<td>16.1 User exits and document flow</td>
<td>490</td>
</tr>
<tr>
<td>16.2 User exits and components</td>
<td>491</td>
</tr>
<tr>
<td>16.2.1 Receivers</td>
<td>492</td>
</tr>
<tr>
<td>16.2.2 Inbound Fixed Workflow</td>
<td>495</td>
</tr>
<tr>
<td>16.2.3 Variable Workflow</td>
<td>496</td>
</tr>
<tr>
<td>16.2.4 Outbound Fixed Workflow</td>
<td>498</td>
</tr>
<tr>
<td>16.2.5 Senders</td>
<td>499</td>
</tr>
<tr>
<td>Chapter 17. Business case and development methodology for</td>
<td></td>
</tr>
<tr>
<td>implementing user exits</td>
<td></td>
</tr>
<tr>
<td>17.1 Business case</td>
<td>502</td>
</tr>
<tr>
<td>17.1.1 Use case</td>
<td>502</td>
</tr>
<tr>
<td>17.1.2 Summary</td>
<td>505</td>
</tr>
<tr>
<td>17.2 User exit development methodology</td>
<td>506</td>
</tr>
<tr>
<td>17.2.1 Requirement gathering</td>
<td>506</td>
</tr>
<tr>
<td>17.2.2 Design</td>
<td>508</td>
</tr>
<tr>
<td>17.2.3 Implementation</td>
<td>511</td>
</tr>
<tr>
<td>17.2.4 Testing methodology</td>
<td>513</td>
</tr>
<tr>
<td>17.2.5 Error handling, problem determination in WebSphere BI Connect</td>
<td>513</td>
</tr>
<tr>
<td>17.2.6 Business case test strategy</td>
<td>518</td>
</tr>
<tr>
<td>17.2.7 Predeployment tests</td>
<td>518</td>
</tr>
<tr>
<td>17.2.8 Deployment</td>
<td>519</td>
</tr>
<tr>
<td>17.2.9 Post-deployment tests</td>
<td>519</td>
</tr>
<tr>
<td>Chapter 18. Implementing receiver and sender user exits</td>
<td></td>
</tr>
<tr>
<td>18.1 Updating WebSphere BI Connect</td>
<td>536</td>
</tr>
<tr>
<td>18.2 The frameworks and building blocks</td>
<td>536</td>
</tr>
<tr>
<td>18.2.1 Transport development</td>
<td>536</td>
</tr>
<tr>
<td>18.3 The receiver</td>
<td>537</td>
</tr>
<tr>
<td>18.3.1 Receiver processing flow</td>
<td>538</td>
</tr>
<tr>
<td>18.3.2 The receiver framework</td>
<td>540</td>
</tr>
<tr>
<td>18.3.3 WebSphere BI Connect provided receiver and custom handlers</td>
<td>551</td>
</tr>
<tr>
<td>18.4 The sender</td>
<td>552</td>
</tr>
<tr>
<td>18.4.1 Sender processing flow</td>
<td>553</td>
</tr>
<tr>
<td>18.4.2 The sender framework</td>
<td>554</td>
</tr>
<tr>
<td>18.4.3 WebSphere BI Connect provided sender and custom handlers</td>
<td>562</td>
</tr>
<tr>
<td>18.5 Deploying and testing the receiver and sender</td>
<td>563</td>
</tr>
</tbody>
</table>
# Figures

1-1 Relationship between B2Bi and e-Marketplaces ........................................ 6
1-2 Data and information sharing B2Bi .......................................................... 14
1-3 Using an internal EAI with B2Bi server ................................................. 16
1-4 B2Bi hub ..................................................................................................... 17
1-5 B2Bi on a business process level ............................................................... 18
2-1 MQ message .............................................................................................. 24
2-2 Secret key cryptography ............................................................................. 27
2-3 Public key cryptography ............................................................................ 27
3-1 Run-time components of WebSphere BI Connect ........................................ 36
3-2 Going from target to gateway ..................................................................... 38
3-3 Stack of B2B formats and standards ......................................................... 39
3-4 Integration of WebSphere BI Connect within an IT infrastructure ............ 41
4-1 Using WebSphere BI Connect Enterprise in a two server environment ...... 46
4-2 Using WebSphere BI Connect Advanced on a single AIX machine .......... 47
4-3 Connecting both WebSphere BI Connect servers via AS2/HTTP ............... 48
4-4 Add encryption and digital signature to exchange of EDI documents ....... 49
4-5 Implementing WebSphere BI Connect Express ........................................... 50
4-6 Adding a new partner and a new document type ....................................... 51
4-7 Integrating an FTP server with WebSphere BI Connect ............................. 52
4-8 Integrating WebSphere Data Interchange with WebSphere BI Connect .... 54
4-9 Routing and look-up in WebSphere Data Interchange ............................... 55
5-1 System architecture for Company E .......................................................... 58
5-2 Verifying the level of WebSphere MQ ........................................................ 59
5-3 Verifying the installed components of WebSphere MQ ............................ 60
5-4 Verifying if SupportPac MA0C is installed .............................................. 61
5-5 Output of the db2level command ............................................................. 62
5-6 Installation options for Microsoft .NET Framework SDK ........................... 63
5-7 Reviewing and changing the DB2 environment ......................................... 64
5-8 Updating the environment variable PATH ................................................ 64
5-9 List of environment variables ..................................................................... 65
5-10 Adding a new user ..................................................................................... 67
5-11 Creating a new group and adding members ............................................. 68
5-12 Create Queue Manager ........................................................................... 69
5-13 Updating the properties of the queue manager ......................................... 70
5-14 Creating a new custom service for the broker ......................................... 71
5-15 Queue manager and its services .............................................................. 72
5-16 Selecting Create the Database from the main installation program ....... 74
5-17 Providing the database system information to the installation program .... 75
5-18 Providing the location information for the tablespaces
5-19 Providing the user ID, password and group information
5-20 Naming the mount point where the common data is stored
5-21 Choosing to run the scripts now or later
5-22 Simple validation process
5-23 Launching the product installation
5-24 Selecting the components to install
5-25 Providing database server information
5-26 Retrieving information from the database server
5-27 Providing MQ connection information
5-28 Providing information for the Community Console component
5-29 Starting WebSphere BI Connect via Windows Services
5-30 Starting WebSphere BI Connect from a command line
5-31 Welcome window for WebSphere BI Connect
5-32 Changing the password during initial logon
5-33 Main window of WebSphere BI Connect
5-34 Document Viewer Search window
5-35 Reviewing the password policy
6-1 Output of the DB2 command db2level
6-2 List of installed components for DB2
6-3 Updating the database instance after installing FixPak
6-4 Verifying the level of WebSphere MQ
6-5 Verifying the installed components of WebSphere MQ
6-6 Searching for the broker start command
6-7 Verifying the availability of a C++ compiler
6-8 WebSphere MQ configuration
6-9 WebSphere MQ configuration
6-10 Updating the queue manager parameters
6-11 Launchpad of WebSphere BI Connect
6-12 Providing database system information to the installation program
6-13 Providing location information for the tablespaces
6-14 Providing user ID, password and group information
6-15 Warning about the C compiler
6-16 Choosing whether to run the scripts now or later
6-17 Simple validation process
6-18 Launching the product installation
6-19 Selecting the components to install
6-20 Providing the database server information
6-21 Retrieving information from the database
6-22 Providing user information
6-23 Providing MQ connection information
6-24 Providing information for the Community Console component
6-25 Simple script file to launch all components of WebSphere BI Connect
<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-29</td>
<td>Connection attributes for AS2: Sending from Company E to A</td>
</tr>
<tr>
<td>8-30</td>
<td>Details of an AS2 communication with encryption</td>
</tr>
<tr>
<td>8-31</td>
<td>Raw document viewer for encrypted document</td>
</tr>
<tr>
<td>8-32</td>
<td>Signing a document by calculating a message digest</td>
</tr>
<tr>
<td>8-33</td>
<td>Verifying the digital signature</td>
</tr>
<tr>
<td>8-34</td>
<td>List of certificates for hubadmin on Company A</td>
</tr>
<tr>
<td>8-35</td>
<td>Uploading the public key of Company E to Company A</td>
</tr>
<tr>
<td>8-36</td>
<td>List of certificates for hubadmin on Company A</td>
</tr>
<tr>
<td>8-37</td>
<td>Updating AS2 connection attributes</td>
</tr>
<tr>
<td>8-38</td>
<td>Uploading the public key of Company A to Company E</td>
</tr>
<tr>
<td>8-39</td>
<td>Update role of public certificate, Company A, on server, Company E</td>
</tr>
<tr>
<td>8-40</td>
<td>Updating the AS2 connection attributes</td>
</tr>
<tr>
<td>8-41</td>
<td>Details of an AS2 exchange with encryption and signatures</td>
</tr>
<tr>
<td>8-42</td>
<td>Encrypted and signed AS2 document</td>
</tr>
<tr>
<td>8-43</td>
<td>Signed MDN</td>
</tr>
<tr>
<td>9-1</td>
<td>Outbound flow within a WebSphere BI Connect Express environment</td>
</tr>
<tr>
<td>9-2</td>
<td>Single machine implementation of WebSphere BI Connect Express</td>
</tr>
<tr>
<td>9-3</td>
<td>Launching the installation of WebSphere BI Connect Express</td>
</tr>
<tr>
<td>9-4</td>
<td>Possible warning issued by the browser</td>
</tr>
<tr>
<td>9-5</td>
<td>Installation summary</td>
</tr>
<tr>
<td>9-6</td>
<td>Online logging in console window</td>
</tr>
<tr>
<td>9-7</td>
<td>Initial logon to WebSphere BI Connect Express</td>
</tr>
<tr>
<td>9-8</td>
<td>Initializing passwords during initial logon</td>
</tr>
<tr>
<td>9-9</td>
<td>Logon using the new password</td>
</tr>
<tr>
<td>9-10</td>
<td>Create Participant (Part 1 of 2)</td>
</tr>
<tr>
<td>9-11</td>
<td>Create Participant (Part 2 of 2)</td>
</tr>
<tr>
<td>9-12</td>
<td>Initial configuration of WebSphere BI Connect Express</td>
</tr>
<tr>
<td>10-1</td>
<td>Scenario overview</td>
</tr>
<tr>
<td>10-2</td>
<td>HTTP settings for partner Company E</td>
</tr>
<tr>
<td>10-3</td>
<td>Updating the HTTP settings related to partner Company E</td>
</tr>
<tr>
<td>10-4</td>
<td>Editing the AS2 settings related to partner Company E</td>
</tr>
<tr>
<td>10-5</td>
<td>Managing My Profile for Company X</td>
</tr>
<tr>
<td>10-6</td>
<td>Creating a new participant</td>
</tr>
<tr>
<td>10-7</td>
<td>Creating a document flow definition at protocol level</td>
</tr>
<tr>
<td>10-8</td>
<td>Creating a document flow at the document flow level</td>
</tr>
<tr>
<td>10-9</td>
<td>Link document flow to packages and protocols</td>
</tr>
<tr>
<td>10-10</td>
<td>Managing XML formats in WebSphere BI Connect</td>
</tr>
<tr>
<td>10-11</td>
<td>Sample XML file (edited)</td>
</tr>
<tr>
<td>10-12</td>
<td>Creating a new XML format</td>
</tr>
<tr>
<td>10-13</td>
<td>Creating an interaction from packaging AS to None for customXML</td>
</tr>
<tr>
<td>10-14</td>
<td>Creating an interaction from packaging AS to None for customXML</td>
</tr>
<tr>
<td>10-15</td>
<td>List of participants at Company E</td>
</tr>
<tr>
<td>10-16</td>
<td>Setting the B2B capabilities of Company X on server of Company E</td>
</tr>
</tbody>
</table>
10-17 Creating a new gateway .................................................. 272
10-18 List of gateways for Company X ................................. 273
10-19 Updating the B2B capabilities of Company E ................. 274
10-20 Connection to send AS2-XML to Company X ............. 275
10-21 Connection to receive AS2-XML documents from Company X ... 275
10-22 AS2 Received Documents Viewer of WebSphere BI Connect Express 277
10-23 Manually sending an XML document ....................... 278
10-24 AS2 Sent Documents Viewer of WebSphere BI Connect Express ... 279
11-1 Scenario overview ......................................................... 283
11-2 Outbound flow ............................................................... 285
11-3 Required directory structure for outbound FTP ............ 286
11-4 Creating a participant profile for Company F ............. 288
11-5 Creating a new target of type FTP Directory ............... 289
11-6 Enabled document flow definitions ................................ 290
11-7 Interaction for None/Binary to None/Binary ............. 291
11-8 Interaction for None/customXML/purchaseOrder for source and target292
11-9 Setting B2B capabilities for Company F ..................... 293
11-10 Creating the FTP gateway within the profile of Company F ... 294
11-11 List of gateways for Company F ................................. 295
11-12 Updating the B2B capabilities of Company E ............. 296
11-13 List of possible connections between Company E and Company F 297
11-14 List of activated connections between Company E and Company F 298
11-15 Using the document viewer for exchanges via FTP ......... 299
11-16 Events associated with the FTP exchange ................. 300
11-17 Contents of the home directory of the user company_e .... 301
11-18 Installing FTP server ................................................. 302
11-19 Control panel for the FTP server ............................... 303
11-20 Directory structure to support inbound FTP ............ 304
11-21 Creating a new server ............................................... 305
11-22 Selecting the new server to configure more properties .... 305
11-23 Specifying the server-wide security settings ........... 306
11-24 Setting the server logging parameters ..................... 307
11-25 User configuration window ................................. 308
11-26 Creating a new user .................................................. 308
11-27 Setting a password for the new user ....................... 309
11-28 Configuring the home directory for user company_f .... 309
11-29 Setting access rights for user company_f .............. 310
11-30 Restarting the server ............................................ 311
11-31 List of possible participant connections .............. 312
11-32 List of activated connections between Company F and Company E 313
11-33 FTP client uploading a binary file to Company E .......... 314
11-34 Using the document viewer for files received via FTP .... 315
11-35 Using the document viewer for XML files received via FTP ... 316
13-4  Components of an EDI message ................................. 372
13-5  EDI VAN network ............................................. 374
13-6  Business initiatives = business integration .................. 376
13-7  The IBM EDI solution .......................................... 379
13-8  Point-to-point EDI solution ................................... 382
13-9  Integration broker EDI solution ................................. 383
13-10 The B2B gateway solution ...................................... 383
14-1  WebSphere Data Interchange development environment .... 386
14-2  WebSphere Data Interchange run-time environment ........ 388
14-3  Installation folder for WebSphere Data Interchange MP V3.2 390
14-4  Create Queue Manager window ................................. 393
14-5  Creating a trigger monitor service component ................ 395
14-6  Creating a new system in WebSphere Data Interchange Client 398
14-7  Connecting to the database .................................... 399
14-8  Mailbox configuration window ................................. 400
14-9  Settings of a Mailbox profile ................................ 401
14-10 Default Network profiles ...................................... 402
14-11 Settings of a Network profile ................................ 402
14-12 MQSeries Queue profiles ..................................... 404
14-13 Details of an MQSeries Queue profile ......................... 405
14-14 Default Service profiles ...................................... 406
14-15 Detailed view of the settings of a Service profile ........ 407
14-16 Common Files tab for a Service profile ...................... 408
14-17 Output Files tab of a Service profile ......................... 409
14-18 Icon to access Trading Partner profiles ...................... 410
14-19 Default Trading Partner profiles .............................. 410
14-20 Settings of a Trading Partner profile ......................... 411
14-21 WDI Proc Options tab of a Trading Partner profile ........ 412
14-22 Mapping editor icons ........................................ 413
14-23 Sample rule for a data transformation map .................. 413
14-24 Details of the usage of a map ................................ 414
14-25 Envelope Attributes tab of the usage of a map .............. 415
15-1  Transformation between EDI, XML for documents from/to Company A 421
15-2  Select transformation based on the destination of the document 422
15-3  Routing incoming documents based on document type ........ 423
15-4  Overview of WebSphere MQ setup ............................. 426
15-5  Creating JMS objects using JMSAdmin ......................... 429
15-6  Creating a new JMS gateway ................................... 431
15-7  Changing the gateway on the participant connection ........ 432
15-8  Creating a JMS target ........................................... 433
15-9  Changing the gateway on the participant connection ........ 434
15-10 Received EDI message ......................................... 435
15-11 Rules, formats header for a message sent by WebSphere BI Connect 436
15-12 Reading from a file and writing to a queue . . . . . . . . . . . . . . . . . . . . . 438
15-13 Populating the RFH header . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 439
15-14 Creating a new XML dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 440
15-15 Select a System . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
15-16 Import XML DTD . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 441
15-17 List of DTDs available in WebSphere Data Interchange database . . 444
15-18 Create a Data Transformation Map: Map Name . . . . . . . . . . . . . . . . 445
15-19 Create a Data Transformation Map: Source or Target . . . . . . . . . . . . 446
15-20 Create a Data Transformation Map: Source Syntax Type . . . . . . . . . . 447
15-21 Create a Data Transformation Map: Source Dictionary . . . . . . . . . . . . 448
15-22 Create a Data Transformation Map: Source Transaction . . . . . . . . . . 449
15-23 Create a Data Transformation Map: Target Syntax Type . . . . . . . . . . . . 449
15-24 Create a Data Transformation Map: Target Dictionary . . . . . . . . . . . . 450
15-25 Create a Data Transformation Map: Target XML DTD . . . . . . . . . . . . 450
15-26 Create a Data Transformation Map: Confirmation . . . . . . . . . . . . . . . . 451
15-27 The map editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 452
15-28 Command Group Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 453
15-29 Command editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 453
15-30 Mapping commands with routing information . . . . . . . . . . . . . . . . . . . 454
15-31 Building a mapping command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 455
15-32 Invoking a built-in function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 456
15-33 Partially complete mapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 457
15-34 Mappings for the element orderHeader . . . . . . . . . . . . . . . . . . . . . . . . . 458
15-35 Inserting a ForEach command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 459
15-36 Creating a local variable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 459
15-37 Mappings for the lineDetail element . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 460
15-38 Creating a new trading partner profile . . . . . . . . . . . . . . . . . . . . . . . . . . 461
15-39 List of trading partner profiles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 462
15-40 Creating a new rule for the map . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 463
15-41 XML document created by WebSphere Data Interchange . . . . . . . . . 468
15-42 Mapping the XML document to an XML document . . . . . . . . . . . . . . . 470
15-43 Mapping the XML document to EDI (Part 1 of 3) . . . . . . . . . . . . . . . . 471
15-44 Mapping the XML document to EDI (Part 2 of 3) . . . . . . . . . . . . . . . . 472
15-45 Mapping the XML document to EDI (Part 3 of 3) . . . . . . . . . . . . . . . . 473
15-46 If-then-else command in the router map . . . . . . . . . . . . . . . . . . . . . . . . . 474
15-47 Building an If command . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 475
15-48 Data transformation rule (Part 1 of 2) . . . . . . . . . . . . . . . . . . . . . . . . . . 476
15-49 Data transformation rule (Part 2 of 2) . . . . . . . . . . . . . . . . . . . . . . . . . . 477
15-50 Creating a translation table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 480
15-51 Using the translate function in a map . . . . . . . . . . . . . . . . . . . . . . . . . . 481
15-52 Sample outbound XML message . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 482
15-53 EDI document after translation and before sending . . . . . . . . . . . . . . . 483
15-54 Mapping EDI 846 to XML . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 485
<table>
<thead>
<tr>
<th>21-6</th>
<th>List of targets on the server of Company E</th>
<th>718</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-7</td>
<td>Creating a JMS gateway in the profile of Company E</td>
<td>719</td>
</tr>
<tr>
<td>21-8</td>
<td>Creating an HTTP gateway in the profile of Company A</td>
<td>720</td>
</tr>
<tr>
<td>21-9</td>
<td>Manage document flow definitions</td>
<td>721</td>
</tr>
<tr>
<td>21-10</td>
<td>Importing a package into WebSphere BI Connect</td>
<td>722</td>
</tr>
<tr>
<td>21-11</td>
<td>Package uploaded successfully</td>
<td>723</td>
</tr>
<tr>
<td>21-12</td>
<td>Verifying the RosettaNet packages</td>
<td>724</td>
</tr>
<tr>
<td>21-13</td>
<td>Receiving RosettaNet P.O. Request Business Messages</td>
<td>726</td>
</tr>
<tr>
<td>21-14</td>
<td>Sending RosettaNet P.O. Confirmation Business Messages</td>
<td>727</td>
</tr>
<tr>
<td>21-15</td>
<td>Company E’s B2B capabilities</td>
<td>729</td>
</tr>
<tr>
<td>21-16</td>
<td>Company A’s B2B capabilities</td>
<td>730</td>
</tr>
<tr>
<td>21-17</td>
<td>Seller’s system: Company E to Company A interaction</td>
<td>731</td>
</tr>
<tr>
<td>21-18</td>
<td>Seller’s system: Company A to Company E interaction</td>
<td>732</td>
</tr>
<tr>
<td>21-19</td>
<td>Using runmqsc to define queues</td>
<td>734</td>
</tr>
<tr>
<td>21-20</td>
<td>Using JMSAdmin</td>
<td>738</td>
</tr>
<tr>
<td>21-21</td>
<td>List of targets on the server of Company A</td>
<td>740</td>
</tr>
<tr>
<td>21-22</td>
<td>List of gateways in the profile of Company A</td>
<td>741</td>
</tr>
<tr>
<td>21-23</td>
<td>List of gateways in profile of Company E on server of Company A</td>
<td>742</td>
</tr>
<tr>
<td>21-24</td>
<td>Verifying the RosettaNet packages</td>
<td>743</td>
</tr>
<tr>
<td>21-25</td>
<td>Receiving RosettaNet P.O. Confirmation Business Messages</td>
<td>745</td>
</tr>
<tr>
<td>21-26</td>
<td>Sending RosettaNet P.O. Request Business Messages</td>
<td>746</td>
</tr>
<tr>
<td>21-27</td>
<td>Company A’s B2B capabilities</td>
<td>747</td>
</tr>
<tr>
<td>21-28</td>
<td>Company E’s B2B capabilities</td>
<td>748</td>
</tr>
<tr>
<td>21-29</td>
<td>Buyer’s system: Company A to Company E interaction</td>
<td>749</td>
</tr>
<tr>
<td>21-30</td>
<td>Buyer’s system: Company E to Company A interaction</td>
<td>750</td>
</tr>
<tr>
<td>21-31</td>
<td>RFHUtil Main tab</td>
<td>752</td>
</tr>
<tr>
<td>21-32</td>
<td>RFHUtil Data tab</td>
<td>753</td>
</tr>
<tr>
<td>21-33</td>
<td>RFHUtil MQMD tab</td>
<td>754</td>
</tr>
<tr>
<td>21-34</td>
<td>RFHUtil RFH tab</td>
<td>755</td>
</tr>
<tr>
<td>21-35</td>
<td>RFHUtil jms tab</td>
<td>756</td>
</tr>
<tr>
<td>21-36</td>
<td>RFHUtil usr tab</td>
<td>757</td>
</tr>
<tr>
<td>21-37</td>
<td>Contents of the usr folder of a RosettaNet message</td>
<td>758</td>
</tr>
<tr>
<td>21-38</td>
<td>Transaction list in the RosettaNet viewer</td>
<td>759</td>
</tr>
<tr>
<td>21-39</td>
<td>Details of a RosettaNet transaction</td>
<td>760</td>
</tr>
<tr>
<td>21-40</td>
<td>Document received by WebSphere BI Connect</td>
<td>761</td>
</tr>
<tr>
<td>21-41</td>
<td>Outgoing RosettaNet document (Part 1 of 2)</td>
<td>762</td>
</tr>
<tr>
<td>21-42</td>
<td>Outgoing RosettaNet document (Part 2 of 2)</td>
<td>763</td>
</tr>
<tr>
<td>21-43</td>
<td>Transaction list in the RosettaNet viewer</td>
<td>764</td>
</tr>
<tr>
<td>21-44</td>
<td>Details of a RosettaNet transaction</td>
<td>765</td>
</tr>
<tr>
<td>21-45</td>
<td>RosettaNet acknowledgment</td>
<td>765</td>
</tr>
<tr>
<td>21-46</td>
<td>Outgoing RosettaNet acknowledgment (Part 1 of 2)</td>
<td>766</td>
</tr>
<tr>
<td>21-47</td>
<td>Outgoing RosettaNet acknowledgment (Part 2 of 2)</td>
<td>767</td>
</tr>
</tbody>
</table>
# Tables

8-1 Certificates on server of Company A ................................................. 226  
8-2 Certificates on server of Company E ................................................. 229  
10-1 Attributes to populate when creating a new participant ................. 258  
10-2 Values to use when creating a protocol document flow ............... 259  
10-3 Values to use when creating a document flow ................................. 261  
10-4 Creating the default gateway for Company X ................................. 272  
15-1 Details of the network profile EDI_XML .......................................... 464  
15-2 Details of the mailbox profile EDI_IN .............................................. 464  
15-3 Details of the mailbox profile XML_IN ............................................ 464  
15-4 Details of the network profile .......................................................... 478  
15-5 Details of the mailbox profile XML_IN .............................................. 478  
15-6 Details of the mailbox profile WDI_OUT ........................................... 478  
15-7 Details of the MQSeries queue profile XML_IN ............................... 479  
15-8 Details of the MQSeries queue profile EDI_OUT ............................... 479  
16-1 Actions provided in the base product .............................................. 497  
17-1 User exit development requirements ................................................... 509  
19-1 Document flow parameters and values .............................................. 600  
19-2 Document flow parameters and values .............................................. 600  
19-3 Document flow parameters and values .............................................. 609  
19-4 Document flow parameters and values .............................................. 609
Examples

7-1  Sample EDI file to be sent to Company A .................................................. 179
13-1  X12 transaction (invoice) .......................................................... 372
13-2  EDIFACT message (Purchase Order) ........................................ 373
14-1  Process definition for WDIAdapter program ................................. 394
15-1  JMSAdmin configuration file .................................................. 428
15-2  Contents of usr folder for a message sent by WebSphere BI Connect 436
15-3  Contents ofusr folder packaged for Backend Integration ................ 437
15-4  DTD for a purchase order .......................................................... 442
15-5  Sample EDI message sent by Company A ........................................ 467
15-6  Audit report of the transformation .................................................. 483
15-7  Contents ofusr folder .......................................................... 484
21-1  JMSAdmin configuration file .................................................. 736
21-2  Updated profile for mqm user .................................................. 737
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about 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 the 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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not 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 contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples 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.

COPYRIGHT LICENSE:
This information contains 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. 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 IBM's application programming interfaces.
Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- Eserver®
- Eserver®
- ibm.com®
- pSeries®
- xSeries®
- z/OS®
- AIX 5L™
- AIX®
- DB2 Connect™
- DB2®
- HACMP™
- IBM®
- MQSeries®
- NetVista™
- POWER4+™
- Redbooks™
- Redbooks (logo)™
- SupportPac™
- ThinkCentre™
- VisualAge®
- WebSphere®

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.
Preface

This IBM® Redbook introduces you to business-to-business (B2B) solutions based on IBM WebSphere® BI Connect for B2B. In Part 1, you learn about B2B technologies and features, architecture and integration options of WebSphere BI Connect.

Part 2 describes the implementation of three editions of WebSphere BI Connect on Microsoft® Windows® and AIX®. Within an environment of four trading partners, you learn step-by-step how to implement various B2B scenarios. Sometimes a trading partner wants to use FTP instead of a more modern communication technique. You learn how to integrate an FTP server with WebSphere BI Connect so that you still have the visibility and manageability of WebSphere BI Connect.

This part also demonstrates how to implement an AS2 exchange of electronic data interchange (EDI) documents and custom XML documents. Plus it shows you how to secure such exchanges via digital signatures and encryption.

Part 3 discusses the integration between WebSphere Data Interchange and WebSphere BI Connect. It shows you how the two products can work together in routing and transforming documents to and from trading partners.

While WebSphere BI Connect has many features, you may still experience a situation where a certain protocol or format is not supported by WebSphere BI Connect. Part 4 discusses how to implement various user exits that allow you to use a custom transport, custom packaging, and custom formats.

Part 5 explains the use of RosettaNet. It describes how you can enable RosettaNet support in WebSphere BI Connect. It shows a way to test this support using standard utilities and viewers in WebSphere BI Connect.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Raleigh Center.

Geert Van de Putte is an IT Specialist at the ITSO, Raleigh Center. He is a subject matter expert (SME) for messaging and business integration, with eight years of experience in the design and implementation of WebSphere Business
Integration solutions. Geert has published several Redbooks™ about messaging and business integration solutions. He has also taught several classes about these subjects. Before joining the ITSO, Geert worked at IBM Global Services, Belgium, where he designed and implemented Enterprise Application Integration (EAI) solutions for clients in many industries. Geert holds a Master of Information Technology degree from the University of Ghent in Belgium.

**Jeff Brent** is an Advisory Software Engineer for the WebSphere Business Integration Competency Center (WBICC). His responsibilities include helping independent software vendors develop integration strategies for WebSphere products.

**Ronan Dalton** is accredited as an Advisory IT Specialist working with the IBM Software Services for WebSphere team. He has been working with both WebSphere Data Interchange and WebSphere BI Connect since their inclusion into the WebSphere Portfolio of software. Ronan co-authored the IBM Redpaper *WebSphere Data Interchange Installation and Configuration*, REDP-3600, and the IBM Redbook *Implementing EDI Solutions*, SG24-6906. He has also published several technical papers that are available on WebSphere Developer Domain and has developed SupportPacs for both WebSphere Data Interchange and WebSphere BI Connect. Ronan holds a Post Graduate Diploma in Computing from Griffith College Dublin (GCD) and is currently studying for his MSc in Computing at GCD.

**Rich Kinard** is a Sr. Consulting IT Architect in the IBM Software Services for WebSphere (ISSW) organization. He is an SME in B2B technologies and has over seven years of experience in designing, developing and implementing B2B, Customer Relationship Management (CRM), and EDMS solutions. He has worked on many initiatives with Internet standards organizations to promote B2B interoperability and was a product/development manager of a successful B2B application prior to working for IBM. He holds a Technical Certificate in Computer Communication Systems from the U.S. Air Force.

**Benjamin Thurgood** is a Technical Specialist for Software Group Services in Sydney, Australia, where he provides consultation and services on WebSphere products. He holds a masters degree in IT from Queensland University of Technology, Australia. His areas of expertise include Java™, object-oriented design, and cryptography.

Thanks to the following people for their contributions to this project:

Lee Gavin
ITSO, Raleigh Center

Dave Mulley
IBM UK
Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

► Use the online Contact us review redbook form found at:

ibm.com/redbooks

► Send your comments in an Internet note to:

redbook@us.ibm.com

► Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HZ8  Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195
Overview of B2B technology and WebSphere BI Connect
Business-to-business concepts

This chapter presents an overview of B2B, including its nature, and its evolution over time. It also discusses the concept of Enterprise Application Integration (EAI), which is commonly confused with or mistaken for B2B. It compares EAI and B2B, by examining the similarities and differences that make them separate concepts and processes.
1.1 Impact of the Internet on the world of business applications

At the beginning of the Internet era, IBM invented the term *e-business* to give a name to a new class of powerful software applications and services that, in its vision, needed to be developed in the following years. This class of applications derives its power from combining the universal access and standards of the Internet with the reliability, security, and availability of existing content, core business processes, and applications.

In simplified terms, e-business refers to the use of Internet technologies to improve and transform key enterprise processes. Most organizations understand this and have begun the transformation from traditional applications to their e-business counterparts. This transformation has begun to Web-enable core processes to strengthen customer service operations, streamline supply chains, and reach existing and new customers.

e-business affects virtually every industry. The pace may vary, but the impact is still being felt. Industry players need to at least consider the changes that e-business will have on their industry in general and their company in particular. Those “out in front” may face more risks, but also harvest the rewards of creating and sustaining some real competitive advantage. Those practicing a wait-and-see strategy may not get locked out of the game, but will at best run with the pack.

Even with the fall of dot-com companies, most companies still recognize the need to at least take steps down the path to becoming an e-business company. Probably one of the best-known applications of e-business is *e-commerce*, which refers to buying and selling activities over a digital medium. However, e-business embraces e-commerce and includes intranet applications.

*Note*: e-business, now referred to a On Demand Business, is a broad concept and can affect nearly all aspects of your business. It is the overall strategy, where e-commerce is an extremely important subset of e-business.

1.2 E-commerce

The world of e-commerce is changing rapidly. Some ten years ago *e-commerce* was mostly defined as participating in an *electronic data interchange* (EDI) initiative. Today, e-commerce means much more than just EDI. It means supporting interactive Web sites; it means enabling the communications with multiple exchanges; it means using XML and the Internet to conduct interactive business-to-customer (B2C) and B2B communications.
Every business’s activities, by definition, pertain to goods and services. Those activities may be divided as to whether they involve consumption, creation, transformation, or provision of goods and services, some combination of these, or the management of such activities. Any of these types of activities may involve external business entities (including customers or consumers). For somewhat historical reasons, special emphasis has been given to B2B activities (those that involve other businesses) and to B2C activities.

This definition does not introduce any technology. The definition of B2B and B2C is first and foremost a business issue. The B2B and B2C classifications pertain to activities by which the business interacts with external entities. Certain types of business, such as pure B2B exchanges, wholesalers, or distributorships, may primarily conduct B2B activities. These businesses derive their revenues from other businesses. All the business activities of such companies either are B2B activities or support them. This has important implications for B2B success. Although it is common to think of B2B as being implemented by public (external) business processes, virtually every private (internal) business process is an essential element to the support of B2B activities.

### 1.2.1 Business-to-consumer

The B2C e-commerce model is a publicly accessible Web site that offers products for sale. It is analogous to a store on the street, where anyone can walk in and make a purchase. A new, unknown customer is known as a guest shopper. The guest shopper has the option of making purchases, after providing some general information about themselves to fulfill the transaction (name, address, credit card, and so on). Most B2C sites encourage users to register and become members. In doing so, the business can establish a relationship with the customer, provide better service, and build customer loyalty.

### 1.2.2 Business-to-business

The B2B e-commerce model refers to an e-commerce store specifically designed for organizations to conduct business over the Internet. B2B applications focus on using the Internet, extranet, or both to improve B2B partnerships and transform interorganizational relationships. The two entities are known to each other and all users are registered. Trading can be conducted directly between buyers and sellers or supported by a third-party (an intermediary) within an e-Marketplace.
There are two styles of B2B:

- Business-to-marketplace-to-business (B2M2B; e-Marketplaces)
- Business-to-business integration (B2Bi)

Figure 1-1 shows the breakdown between e-Marketplaces and B2Bi based on the number of buyers and sellers. In the case of B2Bi, there is usually a one-to-one relationship between buyers and sellers. Any other relationship, such as one-to-many, many-to-many, or many-to-one, would fall into the e-Marketplace category.

Each of the e-Marketplace categories has its own unique characteristics, which are reflected in their implementation. For example, in the case where there is one seller and many buyers, the interaction is similar to that of a traditional auction. Therefore, the digital model has to allow for multiple disclosed bids.

**Business-to-business integration**

B2Bi covers programmatic links between arm-length businesses, between companies where a trading partner agreement may be appropriate. A good example of this is supply chain applications or trading partners that engage in some exchange.
The remainder of this book focuses on B2Bi. For simplicity in this redbook, from here forward, we call B2Bi simply B2B.

**Business-to-marketplace-to-business**

The second style covers the e-Marketplace where the model supports B2M2B. The M represents the e-Marketplace, which supports multiple buyers and suppliers. The buying function may be performed online or programmatically.

The traditional B2B model, centered around the buyer-seller transaction paradigm, shows its limitations. It is definite in scale and displays only partial efficiency in terms of market economics. B2M2B overcomes these limitations and leverages existing B2B applications and technology. The e-Marketplace or online trading communities assist multiple buyers and suppliers to exchange information and transactions.

Trading communities are Internet-based hubs that focus on specific industry verticals or specific industry processes. They use various market making mechanisms, such as auctions, exchanges, and aggregation, to mediate any-to-any transactions among businesses. Through the trading communities, hubs, buyers, and sellers can trade electronically with established partners and at the same time gain access to new markets and new parts of the supply chain.

e-Marketplaces can be a public, interactive buying and selling community. Here all members participate in the open. Or, they can be private, invitation-only communities whose members participate in special pricing arrangements or product and service offerings. Online trading communities have the potential to create excellent and efficient markets.

### 1.2.3 Evolution of the B2B data structures

The structure of information exchanged electronically between businesses has evolved over time. This evolution was basically an evolution to support more open and global standards, so that any business could perform electronic document exchanges with any other business.

**First era: National and industry-oriented EDI data structures**

Some twenty years ago, many companies along vertical industries participated in defining the first standards for exchanging information throughout a supply chain. Since this early work started along vertical industries, the standards that were created focused on industries as retail, transportation, automotive, and so on. Besides, this kind of work was occurring in different countries, resulting in the standards which had a vertical industry orientation as well as geographical characteristics. The result was the overlapping of data structures across multiple
industries in different geographies with no interoperability among these standards.

**Second era: International EDI data structures**

The proliferation of multiple EDI standards dramatically increased the implementation cost of EDI. Cross-industry players, such as transportation companies, found themselves having to learn and implement different EDI data structures depending on the industry they served and the region in which they operated. Standards that became the dominant format for conducting e-commerce are ANSI X12 in North America; TRADACOMS in the United Kingdom; GENCOD in France (retail); Uniform Communication Standard for the U.S. grocery industry; and VDA SEDAS in Germany.

Gradually, the different industry groups came together to create one international data standard: Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT). Nowadays, most trade between companies in two different countries is based on the EDIFACT standard. The migration from industry-oriented or nation-oriented standards to international standards is still happening and may take years. However, the EDI world has achieved one common data structure, which will help drive the costs of EDI much lower.

**Third era: National and industry-oriented XML data structures**

During the last twenty years, the majority of EDI growth supported the business focus on direct material procurement and the movement of goods. EDI data structures worked well for direct material purchases given the structured nature of the procurement process. Prices, contracts, delivery, shipping, and many other details are negotiated and determined during the procurement process.

When newer technologies arose and the Internet became the platform for e-commerce, XML appeared on the horizon. XML has offered one key advantage. That is data streams can now be interpreted and presented to both humans and machines. The availability of XML meant that users with a browser could receive data structures without any change to that data’s construct. XML offers other benefits. Since the technology is extensible, users can add data tags regardless of whether the receiving user is prepared to handle the additional data. However, adding extra tags can cause interoperability issues.

In the late 1990s, lacking any XML data structure organizations, vertical groups again led the charge to create specific vertical-industry data structures. The first group to complete this task was RosettaNet, which defined a series of XML data structures for the high-tech supply chain. The lack of already agreed-upon XML data structures and the perceived need to create them rapidly led the vertical groups to create, publish and endorse their own data structures. Examples are xCBL from Commerce One and cXML from Ariba.
Early cross-industry adopters of XML data structures find themselves in a similar situation to early adopters of EDI. They must still learn and implement multiple XML structures, which now include the proprietary ones mentioned above as well as several others. There is no interoperability among the differing XML data structures, and companies need to implement these multiple XML standards to reach all of the constituents in their supply chain. Supporting multiple XML standards will drive the same interoperability issues that existed with EDI during the 1980s and will similarly lead to higher implementation costs.

**Fourth era: International XML data structures**
XML is now being adopted for B2B e-commerce on a national basis and the international use of XML is just a step away. In the meantime, the same issues arise that historically accompanied the lack of an accepted standard. Industry groups, such as RosettaNet, find themselves making changes to accommodate international needs. Meanwhile vendors who have defined their own XML standards want their standards to become international.

Some participants in the e-commerce community acknowledge the role played by e-business XML (ebXML) as one of the early pioneers of a global standard industry set of XML standards. The ebXML work is a joint effort between the United Nations body for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS). UN/CEFACT and OASIS have end users, defining the documents. They both have previously worked in the e-commerce-standards arena. This ensures that strong data dictionaries exist, as do processes for change control, communication, and documentation regarding the standards.

### 1.2.4 Evolution of B2B data communications
In addition to an evolution in the structure of exchanged documents, there has been an evolution in the communication method to exchange documents.

**First era: Point-to-point direct connections**
Early interenterprise computer-to-computer data exchange moved data via primitive computer protocols. At that time, numerous different communication protocols could be used. Some are still found in vertical-industry implementations, such as the BiSync 2780/3780 no-logon protocol of the Uniform Communications Standard for the U.S. grocery industry. Other such early protocols still in use include ANSI Clear and X, Y and Z Modems.

**Second era: Value-added networks**
To communicate across a supply chain with different suppliers, an early EDI user needed to manage a variety of protocols. Implementing and supporting the many
and different communications protocols that were proliferating had a cost. Companies often needed to buy multiple products. These products, in turn, required their own operational assistance with scheduling transmissions, executing the programs and setting up audit and error-handling procedures.

Soon, users migrated to value-added networks (VANs) to resolve this issue. The VAN became popular because it could insulate the protocol of a given trading partner from the protocols used by all the others. The VAN offered protocol conversion and insulated one company from the other so neither company logged onto the other's system. This insulation provided security and eliminated the need to build an operational communications infrastructure capable of supporting communications sessions with multiple concurrent users. The VAN also insulated users from having to synchronize communications. Companies were free to bring down their systems and perform maintenance without coordinating their activities with multiple trading partners.

**Third era: Internet**

In the 1990s, the Internet became the primary focus for conducting e-commerce. The prevailing view was that the Internet would replace the VAN as a network intermediary. Leaving the VANs, early Internet adopters hoped that it would become the universal protocol and would offer unlimited, inexpensive reach. Unfortunately, these early adopters are facing a familiar situation. Again they have had to learn how to deal with differing format protocols, such as FTP, HTTP, and SMTP.

First users have also had to learn how to deal with securing the data during transport. Security concerns have led to the management of certificates for using encryption methodologies such as Secure Sockets Layer (SSL) and Secure/Multipurpose Internet Mail Extensions (S/MIME). Given the absence of prevailing format protocols, vertical industry groups again moved in to sort out confusion by endorsing a given protocol format and encryption methodology.

### 1.3 Enterprise application integration and B2B

A commonly used definition for EAI is the integration of multiple, independently developed, and maintained applications that use incompatible technologies and that are deployed on a wide range of platforms. EAI capabilities for integrating existing and new applications are fundamental for reacting to business change. At its simplest, EAI enables the transfer of information between applications. But EAI can offer so much more. It can automate the flow of data between the applications that make up the business process flow. The applications in the flow must be enabled to send, receive and work with this data, and to return appropriate results.
With this definition, EAI is transformed from the relatively simple coupling of applications to a global business process implementation and integration. Important business processes comprise many applications combined into complex business tasks and they need to be up and running 24x7.

A major characteristic of EAI is automation. The integrated applications and business processes should run completely without human intervention. If human intervention is required, a workflow manager can generate work items to allow a company's staff to participate in the business processes.

We can differentiate three EAI levels. This is an important aspect of the IBM business integration strategy. The strategy positions the WebSphere Business Integration family of products in three tiers:

► The lowest level of EAI is sending information directly from one application to another, for example, using WebSphere MQ messaging. This is known as point-to-point application integration architecture. Logic and formatting for data exchange is totally within the applications.

► Where there are many applications involved, information needs to be intelligently routed to where it is needed. It may have to be transformed or reformatted, for example, using WebSphere BI Message Broker or WebSphere InterChange Server. The connected applications do not need to know about required data formats, but the business logic that starts the exchange is still inside the calling application. This represents the hub-and-spoke application integration architecture.

► At the highest level of integration, a business process is represented by applications started by specific business conditions (or business logic). Here, a workflow manager, such as WebSphere MQ Workflow, supports sophisticated multistep process flows. WebSphere MQ Workflow also works with applications, while allowing human intervention. There is no need for the applications to know anything about the overall business process, which is defined in a business process model. The business process, controlled and executed by a workflow manager, knows when each application has to be invoked and what data is required. The applications do not link to each other. They send results back to a workflow manager. Then the workflow manager uses the business process definition to decide what happens next.

The WebSphere Business Integration family offers the unique advantage that all three levels may be easily combined within an overall solution, based on their common foundation on WebSphere MQ messaging and XML.
1.4 B2B integration

As described earlier, B2Bi describes e-commerce where the relationships between businesses are one-to-one (Figure 1-1 on page 6).

Nowadays, business depends more and more on strategic relationships with suppliers and partners to establish value chains that provide a competitive advantage. B2B integration is application integration extended outside a single company. It is about companies which trade with partners and suppliers over the Internet in real-time. It is about using middleware technologies, such as distributed objects, remote procedure calls, message queueing, data transformation, and publish/subscribe, to connect different applications with the added complication of getting through firewalls. It is about using the Internet to share data across company boundaries. It is about agreeing on a data structures (standards) and exchanging data electronically using these standards.

B2Bi is becoming extremely important in many business areas, for example:

- Financial transactions such as checking account balances, transferring payments, and obtaining credit information
- Manufacturing activities such as supply chain planning and execution
- Retail activities such as checking suppliers stock, placing replenishment orders, and paying suppliers automatically
- Travel tasks such as checking flight, car, or room availability, and making or changing reservations

An IT infrastructure for automating and coordinating B2B processes is clearly necessary for B2Bi. B2Bi improves performance by supporting key principles of business success:

- Faster time-to-market with new products and services
- Better sales process
- Better service
- Lower operational and production costs
- Lower inventory costs.

Implementing B2Bi solutions that span many and different independent organizations is challenging due to the following considerations:

- Heterogeneous data or information
  
  Different applications and users represent information in different ways or use different kinds of information for the same task. Bridging the associated syntactic and semantic gaps in information can require a mixture of transformation capabilities and neutral information representations.
Heterogeneous systems

Information systems at different organizations within the enterprise are composed of various applications. These include Enterprise Resource Planning (ERP) systems, legacy business applications, advanced planning, product data management, document management, and Web-based intranet applications. Organizations also use various middleware technologies, such as messaging and groupware systems, and distributed object frameworks, such as Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). B2Bi solutions must interoperate with these systems.

Heterogeneous business processes

Different businesses do things differently. The internal processes used for handling orders or managing production and planning are often unique to the organizations that deploy them. It is always a challenge to reach agreement on processes that involve multiple organizations.

Dynamic business and technology environment

Besides heterogeneity, B2Bi is characterized by frequent change. Business processes and the internal systems environment change often. Interorganizational agreements are also subject to change. Coordinating these changes across organizations is a difficult task.

Security and reliability of communications

Before two systems between two different organizations can interact, reliable, secure communications pathways must exist. When the pathway includes an open network, such as the Internet, security is even more important.

Organizational autonomy

Approaches to interenterprise processes must respect organizational autonomy and minimize the complexity of mutual commitments among different organizations.

1.4.1 Types of B2B integration

The following sections distinguish three major types of B2B integration:

- Data/applications sharing
- Document exchange
- Process integration

Data/applications sharing

The data/applications sharing type of B2Bi solution makes a set of data/applications available for direct access by outside organizations. This is usually done through an organization’s Web FTP server or messaging
middleware, such as WebSphere MQ. Multiple URLs are dedicated for this purpose and let outside companies access information in the form of HTML, XML or some agreed proprietary form. Returned information can be static (come from the internal database) or invoke execution of the internal enterprise applications (see Figure 1-2). This approach works well for simple interactions. The main advantage is that it requires almost no specialized software or hardware investment from participants.

This approach provides simplicity and speed-to-market. These advantages stem from support of well-known Web technologies to access existing enterprise databases and applications.

Data/applications sharing is about exposing data through the Web. This data is defined by analyzing the existing information within the enterprise and deciding which part is of interest to business partners, providers, or suppliers.

It is also necessary to determine where this data originates from. It can come from either existing enterprise databases or applications. The data format is another important component which needs to be identified. It determines how
information is structured, including the properties of the data elements within that structure. When the enterprise is internally integrated, EAI is usually an application that has to be accessed.

**Document exchange**

The document exchange approach represents the most common current practice for interactions between businesses. Each participant defines an entry point or entry points through which different types of documents (such as EDI documents or XML documents) can be delivered. The difference between this approach and data/applications sharing is that this is a push versus pull communications style, providing more controlled timing for the information exchange. An enterprise with information “pushes” this information to all interested parties. There are two common implementations of this approach: EDI messaging and Web messaging.

For EDI messaging, a VAN service provider delivers messages to entry points into the enterprise and mediates interactions. Bandwidth for EDI networks is expensive, even today. This is why the creators of EDI were mainly concerned about the size of their messages. EDI messages are compressed and use codes to represent complex values. All the metadata is stripped from an EDI message, which makes it difficult to read and debug. The complexity of EDI makes EDI programmers hard to train and expensive to keep, which makes EDI applications expensive to buy and maintain.

An alternative approach to building document exchange is to use XML-based message formats. The communication occurs through the public Internet, rather than VANs, using HTTP or some proprietary messaging protocol such as WebSphere MQ or Java Message Service (JMS) to achieve assured message delivery. Because the public Internet is essentially free, message size is not a factor. This is why XML messages are rich in metadata, making them easy to read and debug. The simplicity of XML makes XML programmers easy to train and less expensive to keep, making XML applications less expensive to buy and maintain. Additional non-functional requirements such as security must also be factored into the design.

For internal support of document data collection, we can use an internal EAI (message-level) or Business Process Integration (BPI; process-level) implementation. If the integration backbone is not in place, the point-to-point integration between B2B applications and applications, which need to be interacted, must be in place.

Document exchange can be implemented using either a standard Web server or a specialized B2Bi server. The advantage of the first approach is cost savings. Although this approach may be a good starting point for the implementation, it is usually not a viable solution for the final system. This is due to the amount of code necessary to implement such features as guaranteed document delivery.
and document data transformation, which are part of any modern B2Bi server. We recommend that you implement a document exchange using specialized B2Bi servers. And if EAI is in place, you can reuse its capabilities of connecting to legacy applications and transforming and routing data to a B2Bi server (Figure 1-3).

![Figure 1-3 Using an internal EAI with B2Bi server](image)

Now, think about application integration. Each time you add a new application to the existing integration infrastructure, its complexity grows. To avoid the interapplication spaghetti phenomenon, in this case, we implement a integration broker. Each application talks with only one integration broker and it is not aware of other applications. That is the transition from point-to-point to hub-and-spoke architecture.
We can make similar conclusions about B2Bi servers. Point-to-point communications problems, where each company needs to make communication links with suppliers and partners, are possible to solve with a B2Bi hub (Figure 1-4).

![B2Bi hub](image)

**Figure 1-4   B2Bi hub**

**Process integration**

This type of B2Bi solution deals with building inter-enterprise business processes, which incorporate existing internal enterprise processes. Process integration is an extension of the document exchange. The communications are still done through document exchange, but this exchange happens within the context of the business process. This is the most advanced B2Bi implementation. It transforms existing, disparate enterprises into a cohesive system of business processes, supporting all the functions required by the extended virtual enterprise.

Process-based B2Bi manages the interaction between multiple enterprises under the umbrella of integrated B2Bi and internal business workflow. Business applications or internal business processes execute major steps in the B2B workflow. On the basis of this, we can divide business processes into private and public.
We recommend that you use a two-level implementation of process-based B2Bi (Figure 1-5). With this approach, public processes are implemented using a B2Bi server and private processes using an internal integration broker.

Figure 1-5  B2Bi on a business process level
The advantages of the proposed B2B integration infrastructure are:

- There is a clean separation of private and public business processes, although private processes support public ones.
- The B2Bi server is responsible for all public processes, business rules, data transformation, and routing between partners and internal systems.
- The integration broker is responsible for all private processes, business rules, data transformation and routing. This provides a central point of control.
- Built-in GUI tools allow for process definitions, data transformation definitions, and business rules definitions.
- B2Bi servers usually have built-in security and support for assured message delivery.

The use of process-based B2Bi addresses B2B integration in virtually all areas. It includes agreed message formats, message sequencing, communications and security protocols, workflow steps and business rules. It also reduces the application code required to execute business processes. In a BPI solution, workflow management provides flexibility in changing the sequence of actions, while message routing and transformation provide the same kind of flexibility in changing the flow and format of communications.

### 1.4.2 Summary

Process-based B2Bi is the ultimate B2Bi implementation. This approach allows a virtual enterprise to formalize and automate the way it does business. This is the most expensive proposition. It requires participating enterprises to be internally integrated first, but it usually provides the greatest benefits.
Chapter 2. B2B technologies and standards

This chapter discusses B2B and the technologies and standards that are typically needed to successfully send information from one business to another. It also discusses the minimum that is required by most businesses. Since each business is different and has different requirements, a case-to-case approach needs to be used when deciding which and how many components are required.
2.1 Requirements for a B2B solution

It seems that most businesses have fairly similar requirements relating to the issues of sending data either between applications or between themselves and their trading partners.

These are the requirements for most B2B scenarios (in all or part):

- **Ability to send and receive data**
  This can be structured or unstructured data across a variety of transport protocols, for example HTTP, FTP, or Java Message Service (JMS).

- **Definable data formats**
  Definable message formats allow businesses to communicate with other businesses, with data that may have to travel between different operating systems and programs written in different languages. Having a definable message format also allows for more of a plug-and-play solution so that other business or applications can participate in this data sharing. Most B2B applications have the business map their data into a generic data object. This allows for easier manipulation of the data and for easier mapping to different business data formats. It allows for a layer of abstraction between the data formats of an individual application. It also allows businesses to “plug in” additional business with minimal configuration.

- **Security of data**
  Security needs to be available from the time the data leaves the sending application until it arrives at the receiving application. This is even more important today with many businesses using their intranet or the Internet as the travel medium.

- **Availability of messaging systems**
  Messaging systems need to have a capability for failover or recovery and continuous operations without losing or corrupting any data.

- **Monitoring and auditing capabilities**
  The ability to monitor the progress of data through the system is required to provide a user with the ability to see the progress of their data and an administrator with the ability to perform problem or fault investigation and resolution. Auditing capabilities are needed to determine what has been sent or received and what partner was involved.

- **Transactional support**
  The decision to commit or back out the changes is taken, in the simplest case, at the end of a transaction. However, given the distributed nature of B2B transactions, the concept of transactional support takes on a whole new
meaning and level of complexity. Coordination of local transactions and synchronization of distributed transactions is a key issue for data integrity.

- **Performance**

  The system should have the ability to scale to handle the growing needs of the business.

Businesses choose the technology that they use based on several criteria:

- In-house skill
- Cost to retrain present employees
- Cost to adopt new technology
- Cost to integrate new technology
- Maintenance cost of new technology

For example, if the business has sufficient Java skills in-house, then using a Java-based technology may not have a huge impact on the time required to implement a project. However, if only COBOL programming skills are available in-house, using a Java solution would require more work and add expense and time for retraining the programmers. Should the company not use a Java solution? This depends on where the company is planning to go with its solution and how accepting the programmers are to this new technology. Changing from one technology to another is not difficult, but it may require learning a new programming style and require using different tools to do the programming.

Companies should choose carefully when adding new functionality and skill to a group. Never choose a technology because it is the new buzz word in the industry.

### 2.2 Some terminology

Here are some examples of the different types of technologies that you will encounter with a B2B solution.

#### 2.2.1 Messaging and queuing

*Messaging and queuing* has been used in data processing for many years. Without queuing, sending an electronic message over long distances requires every node on the route to be available for forwarding messages. Also the addressees must be logged on and conscious of the fact that you are trying to send them a message. In a queuing system, messages are stored at intermediate nodes until the system is ready to forward them. At their final destination, they are stored in a queue until the consumer of the message is ready to retrieve them.
Even so, many complex business transactions are processed today without queuing. In a large network, the system may be maintaining thousands of connections in a ready-to-use state. If one part of the system suffers a problem, many parts of the system become unusable.

In message queuing, a *message* is simply a collection of data sent by one program and intended for another program.

Queuing is the mechanism by which messages are held until an application is ready to process them. Queuing allows you to:

- Communicate between programs, which may be running in different environments, without writing the communication code.
- Select the order in which a program processes messages.
- Balance loads on a system by arranging for more than one program to service a queue when the number of messages exceeds a threshold.
- Increase the availability of your applications by arranging for an alternative system to service the queues if your primary system is unavailable.

A message itself normally consists of two parts (see Figure 2-1):

- Control information, which contains such information as:
  - The type of the message
  - An identifier for the message
  - The priority for delivery of the message
  - Whether a response is required

- Application data, for example:
  - Business message type, such as purchase order or shipping notice
  - Business identifiers, such as sender and receiver

![Figure 2-1  MQ message](image)
2.2.2 Electronic data interchange

Electronic data interchange (EDI) is the direct computer-to-computer transfer of business information between applications using a standard message format.

**EDI over value-added network**

EDI over value-added network (EDI/VAN) is a private third-party network. It usually has built-in security features that help protect against unauthorized access to customer data. It is 99.9% available and usually has an archive capability for data copies.

It is secure and reliable, but more expensive than the Internet.

**EDI over the Internet**

EDI over the Internet (EDI-INT) is the transmission of EDI over the Internet. The main purpose of EDI-INT is to reduce the cost of transmission. Two main message transmission standards are used for EDI-INT:

- **AS1**: Uses MIME and SMTP
- **AS2**: Uses MIME and HTTP for process-to-process real time EDI

**Message format standards**

The following message format standards are used:

- **ANSI X12**
  American National Standards Institute committee X12 (ANSI X12) defines data which is separated by characters. The message is organized into documents called *Transaction Sets*. These Transactions Sets are in groups called *Functional Groups*, which are then “wrapped” in an envelope called an *Interchange*.

- **EDIFACT**
  Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT) defines which data segments are mandatory or optional, and the number and order of elements.

- **Other formats include**:
  - United Nations Trade Data Interchange (UNTDI) Standards
  - Organization for Data Exchange through Teletransmission in Europe (ODETTE)
  - Healthcare Information Portability and Accountability Act (HIPAA)
  - Voluntary Inter-industry Communications Standards (VICS)
  - Verband Deutscher Automobilhersteller (VDA)
  - Universal Multi-Octet Coded Character Set (UCS)
2.2.3 Transport protocols

Three transport protocols are used mostly when transferring documents in a B2B solution.

HTTP
HTTP is the de facto standard for transferring World Wide Web documents. This protocol operates over Transmission Control Protocol (TCP) connections, usually over port 80. An HTTP client sends Get, Post, or Head messages to an HTTP server. This allows the exchange of data and resources, such as a URL or file, for example:

- **GET:**
  
  ```
  GET /path/to/file/index.html HTTP/1.0
  ```

- **HEAD:**
  
  Similar to a GET but returns the response header only

- **POST:**
  
  Used to send data to the server

FTP
Also called “Fetch”, FTP requires a client and a server. The client connects to the server and may have permission to do everything that can be done locally on the server, except create new files from scratch. However, FTP is mainly used for uploading and downloading large groups of files at one time.

SMTP
SMTP is the Internet standard host-to-host mail transport protocol. It is traditionally over TCP on port 25. SMTP uses a request-response protocol. Since SMTP is limited in its ability to queue messages at the receiving end, Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP) is used to save the message in a server mailbox.

2.2.4 Security

Security technologies are involved at several layers in a B2B solution. They simply protect access to a resource as well as make a resource unreadable for parties not involved in the interaction.

Access control list
Access control list (ACL) specifies a set of rules regarding who is allowed to access a particular resource.
Encryption
The main use of encryption is to assure the confidentiality of an exchanged document.

- Public Key Infrastructure (PKI)
  With PKI, the encrypting and decrypting functions are comprised of mathematical algorithms and the keys are represented by numbers.

- Secret key cryptography
  With secret key cryptography, also known as symmetric key encryption, one key is used to encrypt and the same key is used to decrypt (Figure 2-2).

![Figure 2-2 Secret key cryptography](image)

- Public key cryptography
  With public key cryptography, there are different keys for encrypting and decrypting functions. In Figure 2-3, something encrypted with key 1 can only be decrypted with key 2.

![Figure 2-3 Public key cryptography](image)

Hashing
Hashing a document is mainly used to protect a document against intended changes or tampering. Recalculating the hash from the received document and comparing it with the received hash value is a technique to discover any changes. You can choose from several algorithms to achieve this.

- SHA-1, 256, 384, 512
  Secure hash algorithm (SHA) takes the message and pads it by adding bits to make it a certain length. It is then parsed into $n$ Mb blocks to make sure that the message is a multiple of 512 or 1024 bits. Next, the hash value is set. This...
is determined by the hash algorithm and by taking the first \( m \)-bits of the fractional parts of the square roots of the \( x \) through \( y \) prime numbers.

**Note:** The values of \( m, x, y \) are determined by the hash algorithm.

- **MD5**
  
  Message digest algorithm 5 (MD5) takes the message and then pads it by adding bits to make it 64 bits shy of being a multiple of 512 bits. Next, a 64-bit representation of the message is added so the message is exactly a multiple of 512 bits. Next, four 32-bit values are used to compute the message digest. This message digest is used to produce the output values.

**Digital signatures and certificates**

With this technology, there is a public and a private key. Either key can be used for encrypting the data, but then only the corresponding key can be used to decrypt the data.

**MIME and S/MIME**

MIME allows messages to contain:

- Multiple objects in a single message
- Text having unlimited line length or overall length
- Character sets other than ASCII, allowing non-English language messages
- Multifont messages
- Binary or application specific files
- Images, audio, video, and multimedia messages

Secure/MIME (S/MIME) adds:

- Message privacy
- Digital signatures
- Tamper detection
- Interoperability
- Seamless integration
- Cross-platform messaging

**Secure Sockets Layer**

Secure Sockets Layer (SSL) is a protocol that was designed to provide secure communications on the Internet. SSL authenticates that the server is “who” it should be. SSL creates a secure communication channel by encrypting all communication between the client and the server. SSL conducts a cryptographic word count (checksum) to ensure data integrity between the server and client. This is the number of bytes in a document, and it is sent along with the encrypted document when the server receives the message.
2.2.5 Java

Java is a programming language based on the concept that software, once written, should run on different kinds of computers, without being rewritten or recompiled. The Java programs are run on Java virtual machines (JVM). Therefore, a Java program written and tested on one platform runs on another platform’s JVM.

2.2.6 Extensible Markup Language

XML has gone from the latest buzzword to an entrenched technology in record time. These days, many businesses are using XML to solve business problems.

XML is an open messaging standard that provides a cross-platform portable mechanism for exchanging data. XML refers to a family of specifications based on a tagged message format for metadata. The tag language has been developed from older markup standards including GML and SGML.

XML definitions for specific business objects, such as messages used by EDI or financial applications, are grouped using “schemas” or document type definitions (DTDs).

The XML standard is fast-growing. It is being adapted to, and supported by, an increasing number of products.

2.2.7 Web services

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked over a network. With Web services, there is a Universal Description, Discovery, and Integration (UDDI) server. On this server, Web services can be located, published and updated. When the desired Web service is located, a Web Services Description Language (WSDL) file is associated with it and contains information about the interface, the implementation and the service provider. With this information, a Web service can be invoked.

The use of a UDDI server is optional. Web service clients can typically retrieve the WSDL from other sources as well.
Introduction to WebSphere BI Connect

The previous two chapters cover a more general discussion about B2B technologies, B2B models, and standards. This chapter considers the B2B product in the IBM WebSphere Business Integration family of products, WebSphere BI Connect. It describes its main features, architecture, and some possible deployment scenarios.
3.1 Features of WebSphere BI Connect

WebSphere BI Connect enables B2B process integration and data sharing between partners of all types and sizes. It is implemented on top of a Java 2 Platform, Enterprise Edition (J2EE), and designed for multitier and single server implementations.

WebSphere BI Connect is provided in three editions:

- **WebSphere BI Connect Express**
  This edition is specifically designed for the small-and-medium business (SMB) market. It has a small footprint and is easy to use, but it has limited features and deployment options.

- **WebSphere BI Connect Advanced**
  This edition has all the features of WebSphere BI Connect. It has a rich set of features and can handle any complexity in your B2B environment. Many more protocols and standards are supported for both inbound and outbound communication. Also, WebSphere BI Connect Advanced has a document engine that can handle the transformation and validation of documents. It also checks for duplicate documents.

- **WebSphere BI Connect Enterprise**
  This edition has the same features of WebSphere BI Connect Advanced. Its main differentiator is that a user of WebSphere BI Connect Enterprise is licensed to implement more connections than WebSphere BI Connect Advanced.

This redbook discusses the use of version 4.2.1 of WebSphere BI Connect Express on Windows, version 4.2.2 of WebSphere BI Connect Advanced on AIX, and version 4.2.2 of WebSphere BI Connect Enterprise on Windows.

3.2 Architecture of WebSphere BI Connect

This section introduces the main run-time components of WebSphere BI Connect. It discusses the configuration process and profile management of WebSphere BI Connect.
3.2.1 Run-time components

The run-time environment of WebSphere BI Connect consists of the following components:

- **Receiver, Document Manager, Community Console**
  These are application servers built on top of WebSphere Application Server Express V5.0.2.

- **Queue manager and publish/subscribe broker**
  These are used to pass around messages and events within the system components and to the outside world.

- **Database**
  This is used to store profile information and state information about ongoing processes in WebSphere BI Connect.

- **Common storage**
  This is usually a large stand-alone storage device where documents are stored during the processing in WebSphere BI Connect and after the processing finishes.

Figure 3-1 on page 36 shows how these components interact.

**Receiver**

The Receiver component is responsible for receiving documents into the WebSphere BI Connect server. This consists of receiving documents from internal applications and from external trading partners. Within the Receiver component, several listeners run that poll targets for documents that need to be processed. Such a target can be a file system directory, a JMS queue, or an HTTP URL, for example.

When a document is received by the receiver, it is copied into the common storage. From here, the Document Manager component picks it up and performs its part of the work. The Receiver informs the Document Manager about any new documents by sending a Java Message Service (JMS) message to it. Except for retrieving configuration information, the Receiver does not interact with the database.
**Document Manager**

The Document Manager is the real workhorse of the system. When it receives a notification from the Receiver about a new file, it performs several functions that can be grouped into the following categories:

- **Document handling or parsing and validation**
  
  This step consists of determining the type of document, validating the syntax, retrieving business identifiers if applicable, and possibly invoking activities such as transformation maps.

- **Document encryption and signing if requested**

- **Packaging**
  
  This involves wrapping the document in a header, trailer, or both. Example packages include AS2 and RosettaNet Implementation Framework (RNIF).

- **Transmission to trading partners**
  
  After retrieving the participant connection that matches with document type and business identifiers, the Document Manager sends the document to the trading partner. It handles retries and time-outs as well.

- **State management**
  
  Every change of state is recorded in the database by the Document Manager, so that it can recover in case of failures.

**Community Console**

The Community Console component is a Web application that allows you to perform configuration work, such as profile management. It also enables you to manage the working system. You can use it to help determine which transactions are in progress and which transactions have failed. The Community Console retrieves most of this information from the database. When using a document viewer in the Console, the Console application retrieves the actual document from the common storage area.

**Queue manager and publish/subscribe broker**

A queue manager has to be created to support communication between the three main run-time components of the product. It is used to pass information about received documents and information about system events for external event and system management.

The publish/subscribe broker is used to publish changes to the configuration to the running systems. For example, if a profile is changed, WebSphere BI Connect publishes a reload configuration message to the subscribing run-time components, such as the Receiver and the Document Manager.
Database
The database is used to store data that can be classified in two broad categories: profile information and state management information. The profile information is largely a read-only category once the system is configured. Changes occur only when new profiles are added or when an existing profile is updated, for example replacing an expired certificate. State management information is a highly active category of information. A single document exchange results in many logging events that contain the state transitions of a document.

The actual document, which can be very large, is not stored in the database. It is always kept in the common storage.

Common storage
The common storage component can be a specific directory for a single server installation or a network file storage (NFS) unit that is attached to all servers that run WebSphere BI Connect components. Documents move in this common storage when their state changes. Also, each document that has ever been processed by WebSphere BI Connect is stored in this common storage area. This allows you to analyze document volumes and retrieve historical information for auditing purposes. Archiving mechanisms exist so that you can offload documents to long-term storage units that are not necessarily connected to the WebSphere BI Connect server that is running.
3.2.2 Configuration components

The configuration components can be divided into three categories:

- **Targets**: The entry-points into WebSphere BI Connect
- **Gateways**: The exit-points out of WebSphere BI Connect
- **Interactions**: The processing between target and gateway

**Targets**

A target is a listening or polling component. For example, you can poll a folder on the file system, a JMS queue, or an HTTP URL for documents coming from business partners or from internal systems. A target is used for both inbound and outbound flows. A target always refers to something that WebSphere BI Connect...
manages and that other resources have to use. A file system target points to a folder where, for example, WebSphere Data Interchange has to deliver an outbound document. An HTTP target points to a URL to which the business partner has to post their documents.

**Gateways**
A gateway is a resource that is used by WebSphere BI Connect to deliver documents to an existing resource. For example, when sending a document to your partner, your own WebSphere BI Connect server needs a gateway that points to the URL that is configured on your partner’s server. Thus, a gateway may point to a remote target. That is a target on the WebSphere BI Connect server of your partner. A gateway can also point to a resource in the internal network, such as a JMS queue that acts as the input queue of WebSphere Data Interchange. Again, such a JMS queue is a resource that exists already, independent of the WebSphere BI Connect server.

**Interactions**
An interaction is anything that happens between the target and the gateway. At the target, WebSphere BI Connect may have received a document via a certain transport and of a certain protocol and of a certain structure. However, the gateway may need a different transport and protocol. An interaction performs the required steps to remove transport headers, unpack the document, repackage the document, and add different protocol headers.

To illustrate this process, see Figure 3-2. Starting from the right, a document may have been received on a file system target. Since it is retrieved from the file system, it is typically not packaged. The document payload is electronic data interchange (EDI), which is validated by WebSphere BI Connect. The outbound gateway is an HTTP gateway along which we use the protocol EDIINT AS1/AS2. Thus, WebSphere BI Connect needs to package the EDI document in AS headers. And since the gateway is an HTTP gateway, HTTP headers need to be added as well. This sequence of actions is called an interaction.

From left to right, we have a similar flow. A document is received on the inbound target, which is HTTP. Thus, it has an HTTP header that needs to be removed. The payload of the HTTP stream has an RNIF structure within which we have a RosettaNet Service Content (RNSC) document. The inbound gateway is a JMS gateway. That means that JMS headers are required, and possibly back-end integration packaging, so that the internal application has sufficient metadata to process the RSNC document.
### 3.2.3 Profile management

Three different types of profiles exist within WebSphere BI Connect. The first profile is a *built-in profile*. It is called Operator, but you can change this name if desired. Members of the profile (or company) Operator have the widest authorities in the system. Usually, the members or users belonging to the company Operator are the system administrators who are responsible for a smooth operation of the server.

A second profile is the *community manager*. This profile is usually reserved for the company that owns the server. Members or users belonging to this profile have a wide range of authorities but are usually not authorized to make system-wide changes. There can only be one community manager for a given WebSphere BI Connect server. However, in a network of interconnected WebSphere BI Connect servers, you have more than one community manager. Usually, every WebSphere BI Connect has exactly one community manager.

The trading partners of the community manager are defined to the WebSphere BI Connect server as *community participants*. As a community participant, you have limited options on the server. For example, you can only use or view objects and documents that relate to you as a community participant.

In a network of interconnected WebSphere BI Connect servers, you may be the community manager of your own server and a community participant on the server of your trading partners. If implemented, you can log on as a community participant to the server of your trading partner. This allows for some level of self-definition and self-management within your trading partner community.
3.3 B2B features and standards

As seen in the previous chapters, many standards are playing a role in a B2B solution. These standards and technologies operate at three different levels:

- **Transport**: How the payload goes from the sender to the receiver
- **Message protocol**: The language of the payload
- **Business protocol**: The meaning of the terms used in the message

Figure 3-3 shows this stack of technologies, above which we list the features of WebSphere BI Connect.

![Figure 3-3 Stack of B2B formats and standards](image)

3.3.1 Transport options

WebSphere BI Connect supports several transport options. Messages can be sent to your partners or received from your partners via:

- HTTP and HTTPS
- FTP
- SMTP
- JMS and MQ

Direct access to the file system is possible as well for internal transport. Since version 4.2.2 of WebSphere BI Connect, it is possible to implement custom transport options.
3.3.2 Messaging protocol options

Three styles of messaging are supported by WebSphere BI Connect. You can use:

- AS1 and AS2
- RNIF 1.1 and RNIF 2.0
- SOAP, to support Web services

Since version 4.2.2 of WebSphere BI Connect, it is possible to implement custom messaging protocol options.

3.3.3 Business document formats

At the business level, WebSphere BI Connect can handle and validate documents that adhere to the following standards:

- RosettaNet 1.1 and RosettaNet 2.0
- EDI X12 and EDIFACT
- cXML
- Custom XML
- Flat files without any formalized structure

3.3.4 Security options in WebSphere BI Connect

B2B transactions over the Internet implies that strong security is required. WebSphere BI Connect has security features at three different levels:

1. Transport level. WebSphere BI Connect supports HTTP/S and secure FTP, so that the exchange of documents occurs in a secure way with authorized partners.

2. Document level. Besides using a secure transport, you can also protect the actual document by using encryption and signing it. WebSphere BI Connect supports using several encryption and signing algorithms. It also has an interface to manage certificates and private keys.

3. Interface level. Using the console is reserved for authenticated users. You need to log on to WebSphere BI Connect via a browser. The browser interface is supported over HTTP and HTTP/S, providing a secure interface to WebSphere BI Connect. As an authenticated user, you belong to a certain group or profile which implies certain authorities. The hub administrator can assign or deny certain authorities so that a user can only perform authorized tasks.
3.4 Integration options

WebSphere BI Connect does not operate on an island within an organization. Documents that are received by WebSphere BI Connect need to be processed by one or more internal applications. WebSphere BI Connect can interact directly with those internal applications via JMS, HTTP, or the file system (Figure 3-4).

For more complex scenarios, it is common for WebSphere BI Connect to interact with an EAI layer, for example, an integration broker such as WebSphere Business Integration Server. The integration broker can then handle the distribution of incoming documents to one or more internal applications. If an incoming document needs to be delivered to only one application, then WebSphere BI Connect can interact with that application directly.

However, if a document needs to be delivered to more than one internal application in a dynamic way (based on the content or type of the document for example), the use of an integration broker will greatly simplify the overall infrastructure. Integration between WebSphere BI Connect and WebSphere
Business Integration Server can be performed via several communication options. Also, the documents can be packaged in a specific Backend Integration packaging that is provided by WebSphere BI Connect. This allows you to exchange detailed metadata between WebSphere BI Connect and WebSphere Business Integration Server so that you can build flexible solutions.

A third type of integration is the integration between WebSphere BI Connect and specialized EDI translators such as WebSphere Data Interchange. The translation and processing of EDI documents is a specialized task that is usually offloaded to dedicated software products. WebSphere Data Interchange can transform the incoming EDI document into XML for example and perform such EDI functions as functional acknowledgments and data validation, to make sure that the received documents adhere to the standard. The transformed document can then be delivered to the internal application by WebSphere Data Interchange.

This last type of integration is the subject of Part 3, “Integration with WebSphere Data Interchange” on page 365. But first let's discuss the implementation of a WebSphere BI Connect infrastructure.
Building a B2B exchange
Chapter 4. Implementation scenarios

This chapter describes the different steps to implement a B2B solution between four business partners. By adding features and functions in a gradual way, it becomes easier to understand why configuration tasks are done. It also offers the implementor a check-point so that he or she can perform some validation before moving to the next step.

This chapter presents a general discussion about the implementation process, and refers you to subsequent chapters for details.
4.1 Implementation of WebSphere BI Connect Enterprise for Windows

The first implementation phase is setting up WebSphere BI Connect Enterprise for Windows in a dual server environment. The three run-time components of WebSphere BI Connect are running on a single machine that has the host name wbichub. The DB2® server and the MQ queue manager are running on a separate machine, with the host name wbicdata.

Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57, discusses details about the planning and decisions that you need to make to build such an environment successfully.

Figure 4-1 shows an overview of this setup. At the end of this phase, you have a working WebSphere BI Connect server, but with no connected partners and no connections with internal systems.

Figure 4-1 Using WebSphere BI Connect Enterprise in a two server environment
4.2 Implementation of WebSphere BI Connect Advanced for AIX

The next phase is the implementation of WebSphere BI Connect Advanced on a single AIX machine. It involves using a local DB2 server and a local MQ queue manager.

Chapter 6, “Implementing WebSphere BI Connect Advanced for AIX” on page 95, describes the setup of all prerequisite components. It also provides tips about how to validate the setup of DB2 or how to verify that prerequisite components are configured correctly. Then, at the end of that chapter, we have a working WebSphere BI Connect server that is not connected to any other partner or system, as shown in Figure 4-2.

![Figure 4-2 Using WebSphere BI Connect Advanced on a single AIX machine](image)

4.3 Implementing a basic exchange

Given the two WebSphere BI Connect implementations on AIX and Windows, you learn how to set up a basic exchange between those two WebSphere BI Connect servers. Chapter 7, “Creating a basic B2B exchange” on page 121, explains how to configure WebSphere BI Connect so that electronic data interchange (EDI) documents can be exchanged in both directions over HTTP in an AS2 package.
Figure 4-3 illustrates the configuration steps to build the environment. A natural approach is to split this into a discussion about the outbound flow and a discussion about the inbound flow and possibly a validation for the outbound flow before proceeding to build the inbound flow. However, this book describes the setup by focusing more on a role-based approach.

In Chapter 7, “Creating a basic B2B exchange” on page 121, we limit the communication to one type of document and to one partner. We also limit ourselves to file-based input and output. That is, we configure WebSphere BI Connect to retrieve EDI documents from a directory and send it to the business partner that is encoded in the EDI document. The target WebSphere BI Connect server drops the EDI document into a directory as well.

Thus, the MQ infrastructure at either side is not used much at this stage, except for communication between WebSphere BI Connect components. At the end of this chapter, you see how to exchange documents in both directions and receive message disposition notifications (MDNs) from the target server.

Figure 4-3  Connecting both WebSphere BI Connect servers via AS2/HTTP
4.4 Securing the exchange between Companies E and A

In Chapter 7, “Creating a basic B2B exchange” on page 121, we assume default values for the AS2 connection. This means that no encryption or digital signatures are used. The EDI document is sent in clear text. This is not good enough for a real-world implementation.

Chapter 8, “Securing the B2B exchange” on page 189, explains how to secure the exchange of EDI documents. It demonstrates how to obtain keys and certificates and how to upload them to WebSphere BI Connect. This is done to switch to an AS2 connection where sensitive business data is encrypted and where EDI documents are signed, so that the receiver can validate their origin.

Chapter 8, “Securing the B2B exchange” on page 189, describes the process again from a role-based perspective, so that it becomes clear which tasks can be performed by either the hub administrator of the WebSphere BI Connect server or by the community participant (Figure 4-4).

![Diagram showing the exchange between Companies A and E](image-url)

**Figure 4-4** Add encryption and digital signature to exchange of EDI documents
4.5 Implementing WebSphere BI Connect Express for Windows

After completing the security setup, we extend the exchange with an additional partner (Figure 4-5). This is again performed in two steps. First, we implement WebSphere BI Connect Express for Windows. Given that WebSphere BI Connect Express has fewer requirements for other software products, such as DB2 and WebSphere MQ, this is a straightforward process, as discussed in Chapter 9, “Implementing WebSphere BI Connect Express on Windows” on page 235.

Chapter 10, “Extending the basic B2B exchange” on page 249, explains how to add another partner to an existing WebSphere BI Connect server. What changes need to be performed to get another partner online? You see how to add a third community participant and another document type (Figure 4-6).

The WebSphere BI Connect Express partner is less sophisticated than the WebSphere BI Connect Advanced partner and wants to exchange simple custom XML documents. Thus, we introduce this new custom XML format to the WebSphere BI Connect Enterprise server of Company E so that it can perform the required routing to Company X, who owns the WebSphere BI Connect Express server. We also contrast the routing logic for WebSphere BI Connect Enterprise (and Advanced) versus the routing logic for WebSphere BI Connect Express.
4.6 Using FTP with WebSphere BI Connect

The B2B exchange is now extended even further, so that it can also support FTP. Company E wants to extend its B2B infrastructure to include a document exchange with Company F. However, Company F is reluctant to jump quickly on a technology that is new to the IT department of Company F. At the same time, Company E wants to build a solution sooner rather than later. A compromise is found between Company E and Company F by implementing an FTP-based solution. Company F has extensive experience with FTP and thus they can implement a document exchange with Company E rather quickly.
Figure 4-7 shows a schematical overview of the complete B2B solution from the perspective of Company E. It shows the existing AS2-based exchanges with Company A (for EDI) and with Company X (for XML). Within this infrastructure, we add an exchange with Company F over FTP, as explained in Chapter 11, “Integrating FTP servers with WebSphere BI Connect” on page 281.

4.7 Managing a B2B infrastructure

Since the infrastructure is now a bit more complex, let’s take a closer look at the manageability of the infrastructure, especially from the perspective of Company E, that has an infrastructure in place with more than one partner. As usual, the change from “1” to “2” introduces another level of complexity, and when this level is understood, moving from “2” to “n” becomes easy.

Each player in the system has an important role. The hub administrator needs to determine if the system as a whole performs well, for example, by ensuring that
there are no bottlenecks or errors. Several tools and options are available to the
hub administrator.

The community manager wants to look at the WebSphere BI Connect system
from a business perspective. This person asks, “What documents have I
received? From whom? What is the status of each document? Did I receive an
MDN for that transaction?” For all of these questions, the WebSphere BI Connect
server contains tools and viewers so that the community manager has a good
view on their business.

In addition to management options for the hub administrator and the community
managers, WebSphere BI Connect also has tools and viewers available for the
community participant. Thus, if the community participant receives no MDN or
receives an error report, they can log on to the WebSphere BI Connect server of
his partner and try to determine why their transaction resulted in an error.

Chapter 12, “Managing the B2B exchange” on page 337, explains all of this in
greater detail.

4.8 EDI translation

The next step in building the WebSphere BI Connect solution infrastructure is to
add WebSphere Data Interchange.

Chapter 13, “Introduction to EDI technology and WebSphere Data Interchange”
on page 367, provides an introduction to EDI and to WebSphere Data
Interchange. Then, Chapter 14, “WebSphere Data Interchange infrastructure and
implementation” on page 385, explains how to implement the WebSphere Data
Interchange Client and Server infrastructure. This includes the creation of an MQ
infrastructure for WebSphere Data Interchange. It also includes the configuration
of MQ communication between the WebSphere Data Interchange machine and
the server that hosts the WebSphere BI Connect database and queue manager.
That is host wbicdata as shown in Figure 4-8.

The next step is to develop transformation maps in WebSphere Data Interchange
so that XML documents can be transformed into EDI documents. This also
requires the need to import definitions for the XML and EDI documents.

When the maps are tested successfully and a correctly built EDI document is in
the outbound queue, you can update the WebSphere BI Connect configuration to
retrieve outbound EDI documents from a queue instead of a file directory.
Assume that you need to send out a purchase order to either Company A or Company X. From a back-office application perspective, it should not matter. The back-office application simply wants to drop such a purchase order in a queue, and the B2B infrastructure as a whole should handle it.

When WebSphere Data Interchange retrieves an XML document from the back-office application and it determines that the target is Company X, then it calls the appropriate map to transform the document from the internal XML structure to the external XML structure that is required for Company X. If WebSphere Data Interchange determines that the document is intended for Company A, then it calls a different map to transform it into the appropriate EDI standard.

The final step is to isolate the internal applications from the business identifiers that are required for AS2. Basically, it is quite common that an internal application refers to a partner with one name or identifier, while a different identifier may be required for B2B communication. Here again, WebSphere Data Interchange can provide a solution by using a look-up table as part of the transformation map (Figure 4-9).
4.9 Extending WebSphere BI Connect

The previous sections describe scenarios where the B2B exchange is extended horizontally. More partners are added using different protocols or document formats. Part 4, "Extending the features of WebSphere BI Connect" on page 487, discusses a vertical way to extend the B2B exchange. This time, we use the exit framework of WebSphere BI Connect to add support for additional protocols and packaging techniques.

WebSphere BI Connect provides support for a number of protocols, packaging techniques, and document styles. However, the world of B2B is using a much wider variety. By providing a user exit framework, customers and solution developers can extend the WebSphere BI Connect product while leveraging its many other features such as community management and transaction visibility.
4.10 Implementing RosettaNet

Part 5, “Implementing RosettaNet in WebSphere BI Connect” on page 691, explains how to implement RosettaNet within WebSphere BI Connect. RosettaNet as an organization has defined standards to outline how documents should be exchanged and packaged and what the vocabulary is that can be used to exchange documents. These three layers match nicely to the three layers that are used within any WebSphere BI Connect document flow:

- Packaging
- Business protocol
- Document type

This part also discusses the implementation of a RosettaNet 3A4 document, which is a purchase order.
Implementing WebSphere BI Connect Enterprise in a Windows environment

This chapter explains how to implement WebSphere BI Connect Enterprise in a Windows environment. If you are mostly interested in an AIX implementation, go to Chapter 6, “Implementing WebSphere BI Connect Advanced for AIX” on page 95, which contains similar information as in this chapter, but that applies to the AIX environment.
5.1 Implementation overview

The three server components of WebSphere BI Connect are going to be installed on the same Windows platform. DB2 and WebSphere MQ are installed on a separate dedicated server. In a later stage, we add WebSphere Data Interchange on another system and connect it via MQ to the queue manager on machine wbicdata that is used by WebSphere BI Connect (Figure 5-1).

5.2 Verifying software levels on the hub and data machine

As for any implementation of a software product, you need to understand the prerequisites, both software and hardware, and you need to know how to verify that prerequisites are met.

WebSphere BI Connect relies on the services and features of two other products:

- A database manager, such as DB2, which is used in this environment
- WebSphere MQ

The following instructions help you to validate that these products are installed at the correct level and that required features are available.
5.2.1 Verifying WebSphere MQ

To verify the version of WebSphere MQ, you can run the program `mqver`. Figure 5-2 shows the output of this program. CSD04 is required for WebSphere BI Connect. Figure 5-2 shows the use of CSD07.

```
C:\>mqver
Name:        WebSphere MQ
Version:     530.7  CSD07
CMVC level:  p530-07-L040527
BuildType:   IKAP - (Production)
```

*Figure 5-2  Verifying the level of WebSphere MQ*

To obtain the required CSD, visit the following Web site, which contains information about the latest CSDs:


WebSphere BI Connect also uses Java Message Service (JMS), which may or may not be installed on a provided computer. To verify whether JMS support is installed, follow these steps:

1. Start the registry editor. Select **Start → Run**.
2. Type: `regedt32`
3. Expand the registry. Select **HKEY_Local_Machine → Software → IBM → MQSeries → CurrentVersion → Components**.
4. In the right pane (Figure 5-3), locate the value JavaMsg.

Note: A default or typical installation of WebSphere MQ does not install JMS support. You need to select the custom installation option.
An additional WebSphere MQ component is required, MQ Publish/Subscribe. This feature is available as a SupportPac™ that carries the identifier MA0C. To verify if a system has this feature installed, you can again explore the registry, as shown in Figure 5-4.

![Figure 5-4 Verifying if SupportPac MA0C is installed](image)


After you download and install this SupportPac, verify the installation in the registry again.

CSD08 of WebSphere MQ installs this component by default, which removes the need to download and install the SupportPac MA0C.

### 5.2.2 Verifying DB2

WebSphere BI Connect requires version 8.1 of DB2 and the installation of FixPak 2. To verify which versions of DB2 and FixPak are installed, open a command window and run the program `db2level`. The output of this program (Figure 5-5) tells you which version of DB2 is installed and which FixPak level is installed.

![Figure 5-5 DB2 level output](image)

DB2 can be installed in many different ways. It consists of several components that you can select or not select during the installation. Since WebSphere BI Connect uses a number of stored procedures that are built during the installation.
of the product, you need to make sure that the DB2 Application Development Toolkit is installed. A good indication of whether the toolkit is installed is to look for the C include files, which are installed in D:\SQLLIB\include.

If this toolkit is not available, add it by running the DB2 installation program one more time. After you add the required components, re-install the FixPak as well.

Figure 5-5 shows the output of the `db2level` program in our environment. It shows that we use FixPak 4.

```plaintext
C:\>db2level
DB21085I  Instance "DB2" uses "32" bits and DB2 code release "SQL08014" with level identifier "02050106". Informational tokens are "DB2 v8.1.4.428", "s040122", "WR21338", and FixPak "4".
Product is installed at "C:\SQLLIB".
```

**Figure 5-5  Output of the db2level command**

### 5.3 Enabling DB2 stored procedures on the data machine

WebSphere BI Connect uses many stored procedures to manage its data. These stored procedures are built during the installation of the product.

While a stored procedure is written in SQL, it results in the creation of a DLL (or loadable module on UNIX®) that is compiled by DB2. This means that a C/C++ compiler must be available on the database server and needs to be configured for use by the user ID that runs the database instance, for example, db2admin.

Several solutions exist for making a C/C++ compiler available to DB2:

- Purchase Visual Studio .NET from Microsoft®.
- Download and install the free GNU C/C++ compiler.
- Download and install the .NET Framework and SDK from Microsoft.

The following sections explain how to install and use the last option in this list.

### 5.3.1 Installing .NET Framework and SDK

To compile the stored procedures, download and install these two packages:

- Microsoft .NET Framework 1.1 Redistributable package
- Microsoft .NET Framework SDK Version 1.1
You can obtain both from the following Web site:

http://msdn.microsoft.com/netframework/downloads/framework1_1/

Or you can link directly to these packages by going to:

- http://download.microsoft.com/download/a/a/c/aac39226-8825-44ce-90e3-bf8203e74006/dotnetfx.exe
- http://download.microsoft.com/download/5/2/0/5202f918-306e-426d-9637-d7ee26fbe507/setup.exe

1. Install the Framework Redistributable package (download dotnetfx.exe).

2. Check to see if you already have the .NET Framework 1.1 installed.
   a. Click **Start** → **Control Panel**.
   b. Double-click the **Add or Remove Programs** icon.
   c. In the Add or Remove Programs window, scroll through the list of applications. If you see Microsoft .NET Framework 1.1 listed, the latest version is already installed.

3. After you install the run-time framework, install the SDK. During the installation of the SDK, you can choose whether to install the samples. In all cases, install the actual SDK, which contains the tools to build applications (Figure 5-6).

![Figure 5-6  Installation options for Microsoft .NET Framework SDK](image)
5.3.2 Configuring DB2 to use the .NET SDK

When the .NET SDK is installed, you must instruct DB2 to use this compilation option. You do this by updating the DB2 environment.

1. Open a DB2 command window.
2. Enter the `db2set` command to change the value of the variable `DB2_SQLROUTINE_COMPILER_PATH`, as shown in Figure 5-7. The C:\Microsoft.NET\SDK\V1.1 directory is where the .NET Framework SDK is installed.

   You can use the `db2set` command without parameters to verify that the variables and values that are currently stored in the DB2 environment.

   ```
   C:\>db2set DB2_SQLROUTINE_COMPILER_PATH=C:\Microsoft.NET\SDK\V1.1\bin\sdkvars.bat
   
   C:\>db2set
   DB2_SQLROUTINE_COMPILER_PATH=C:\Microsoft.NET\SDK\V1.1\bin\sdkvars.bat
   DB2ACCOUNTNAME=WBICDATA\db2admin
   DB2INSTOWNER=WBICDATA
   DB2PORTRANGE=60000:60003
   DB2INSTPROF=C:\SQLLIB
   DB2COMM=TCPIP
   ```

   Figure 5-7   Reviewing and changing the DB2 environment

3. Update the system environment variable PATH.
   a. Right-click **My Computer** and select **Properties**.
   b. Select the **Advanced** tab and click **Environment Variables**.
   c. Locate the variable PATH and click **Edit**.
   d. In the Edit System Variable window (Figure 5-9), select the variable PATH.
   e. Add the C:\Microsoft.NET\SDK\V1.1\bin directory when C:\Microsoft.NET\SDK\V1.1 is the installation directory of the .NET Framework SDK (see Figure 5-8).

   ![Edit System Variable](image)

   Figure 5-8   Updating the environment variable PATH
f. Click **OK**.

Figure 5-9 shows the updated variable.

![Environment Variables](image)

**Figure 5-9  List of environment variables**

4. Restart the DB2 services for them to use these values.

### 5.4 Installing the software for the data machine

This section explains how to install and configure the database component of WebSphere BI Connect. This includes:

- Adding users and groups
- Configuring WebSphere MQ
- Installing the database loader and creating the database

At the end of this section, some simple validation is done to make sure that the database is created and loaded correctly before installing the WebSphere BI Connect product on the hub machine.
5.4.1 Adding user IDs and a group

WebSphere BI Connect uses four different user IDs and one group to manage security roles. The names of these user IDs and group are free to choose. However, we recommend that you use default names. On the database server, only three users are needed. The user bcguser is needed on the WebSphere BI Connect server itself.

**Note:** The server where the user IDs need to be defined, depends on the setup of the Windows security domain and the type of database authentication that is implemented. In our environment, we work in a Windows workgroup with locally defined user IDs and not in a Windows domain where user IDs are defined centrally. Also, the database authentication is server-side.

To define the user IDs, follow these steps:

1. Right-click **My Computer** and select **Manage**.
2. Expand the tree structure in the left pane and select **System Tools → Local Users and Groups → Users**.
3. Right-click the folder **Users** and select **New User**.
4. In the New User window (Figure 5-10), complete these items:
   a. Name the user bcgcon.
   b. Provide a password.
   c. Deselect the User must change password at next logon option.
   d. Select the Password never expires option.
   e. Click Create.
   f. Repeat steps a through e to add users bcgdoc and bcgrecv.

![New User window](image)

Figure 5-10  Adding a new user

5. Expand the tree structure in the left pane and select System Tools → Local Users and Groups → Groups.

6. Right-click the folder Groups and select New Group.
7. In the New Group window (Figure 5-11), complete these tasks:
   a. Name the group bcggroup.
   b. Click Add to add members to this group.
   c. Now in the Members box, you see the list of defined users on this computer. Select the following users: bcgcon, bcgdoc, and bcgrecv.
   d. Click Create.
   e. Click Close to return to Computer Management window.

![Figure 5-11 Creating a new group and adding members](image)

### 5.4.2 Configuring WebSphere MQ

Create and configure some MQ resources for use by WebSphere BI Connect.

1. Start WebSphere MQ Explorer. Select Start → Programs → IBM WebSphere MQ → WebSphere MQ Explorer.
2. Expand the tree in the left pane and right-click the folder Queue Managers. Select New → Queue Manager.
3. In the Create Queue Manager (Step 1) window (Figure 5-12), complete these tasks:
   a. For Queue Manager name, type partner_e.bcg.queue.manager.
   b. For Dead Letter Queue, type SYSTEM.DEAD.LETTER.QUEUE.
   c. Click Next.

![Create Queue Manager (Step 1)](image)

**Figure 5-12  Create Queue Manager**

4. Customize the size of the MQ transaction log. Two types of logs exist, circular and linear. A *circular log* provides a transactional log, but is easier to manage. A *linear log* is required to make and restore backups within WebSphere MQ and to recover from hard disk failures, assuming that messages and log are not stored on the same disk.

   The logging can be spread over multiple log files, and the size of the log file can be configured as well. To make the log a little bigger than the default size, set the log file size to 1024, which is 1024 pages of 4 KB. Click Next.

5. Select the option to create a channel for remote administration. Click Next.

6. Select the option to create a listener for TCP/IP and to use an unused port, for example 9999. Click Finish.

7. When the queue manager is created, start WebSphere MQ Services. Select Start → Programs → IBM WebSphere MQ → WebSphere MQ Services.

8. Expand the tree structure in the left pane to locate the new queue manager. Right-click it and select Properties.
9. In the Properties window (Figure 5-13), complete these items:
   a. Select the **Channels** tab.
   b. Increase MaxChannels and MaxActiveChannels to **1000**. This change ensures that WebSphere BI Connect can establish sufficient MQ client sessions when required.
   c. Click **OK** to return to the MQ Services application.

![Figure 5-13 Updating the properties of the queue manager](image)

In addition to a queue manager, you also need a broker. This broker is used for components of WebSphere BI Connect that use publish/subscribe messaging. To make sure that this broker is started at the same time as the queue manager, add the broker as a custom service to the service that represents the queue manager.

1. In the MQ Services application, right-click the folder **Custom Services** for the queue manager partner_a.bcg.queue.manager and select **New → Custom Service**.
2. In the Create Custom Service window (Figure 5-14), complete the following tasks:
   a. For Service Name, enter a name for the new service, for example, Publish/Subscribe Broker.
   b. Enter the Start and End commands for this service.
      For the Start command, enter:
      `strmqbrk -m partner_e.bc.g.queue.manager`
      For the End command, enter:
      `endmqbrk -m partner_e.bc.g.queue.manager`
   c. Set Execution to **COMMAND**.
   d. Click **OK**.

*Figure 5-14  Creating a new custom service for the broker*
You return to the MQServices application, which now has a new custom service for the queue manager partner_e.bcg.queue.manager (Figure 5-15).

Figure 5-15  Queue manager and its services

You must enter two more commands to configure the MQ resources.

1. Open a command window and change to the folder \Tools\MQSeries on the installation CD. Type:

   runmqsc partner_e.bcg.queue.manager < create_wbic_queues.mqsc

2. Change to the directory \Java\bin within the WebSphere MQ installation directory and enter:

   runmqsc partner_e.bcg.queue.manager < MQJMS_PSQ.mqsc

The first command defines the number of MQ resources used by WebSphere BI Connect. The second file is installed as part of the Java Messaging feature of WebSphere MQ. It contains MQ definitions used by JMS Publish/Subscribe.
5.4.3 Installing the database schema

Besides WebSphere MQ, WebSphere BI Connect relies on the services of a database manager. For this redbook, we used DB2. However, other database products are also supported. The database is used to store data that can be classified into two broad categories: profile information and logging information.

To simplify, the profile information is largely a read-only category when the system is configured. Changes occur only when new profiles are added or when an existing profile is updated, for example replacing an expired certificate.

The logging information is a highly active category of information. A single document exchange results in many logging events, which is useful in keeping track of what happens in the system or what has happened in the system. Thus, this category of information expands constantly and is consulted mostly by online users and administrators.

Given the different nature of the database access to both categories of information, it comes as no surprise that WebSphere BI Connect has grouped them into two different database tablespaces so that it becomes easy to manage them separately. This can mean storing the information on separate disks and implementing different backup and reorganization procedures.

1. Start the product installation program from the product CD.
2. You see several options. The two main tasks are:
   - Create the Database
   - Install the Product

Select the Create the Database option as shown in Figure 5-16 to launch the database loader program. Be aware that the database loader program creates a database that is owned by the currently logged-on user. If you want the database to be owned by the database instance owner (for example, db2admin), log on to the system with this user ID.

The database loader program does not perform a context switch, even though you may think it is going to do so, based on the information that you provide (see Figure 5-17 on page 75).
3. When the database loader program is started, the Welcome window opens. Click **Next**.

4. Accept the software license. Then click **Next**.

5. Provide an installation directory (for example, `C:\WBIC\Connect\DBLoader`).

6. Select the database server product that is going to be used. This can be:
   - IBM DB2 8.1.2 or later
   - Oracle 9i 9.2.0 or later

   The following figures show the choices that we made for our environment and are specific for DB2. The product documentation explains this for Oracle as well.
7. In the Database Loader window (Figure 5-17), enter the following information:
   a. Enter the name of the new database for use by WebSphere BI Connect, which by default is BCGAPPS.
   b. Type the name of the database instance. A standard Windows installation of DB2 has a default instance called DB2. Other instances may also be created. This instance must exist before you run the database loader program.
   c. Specify the name of the group to which the database owner user ID belongs. On Windows, this is usually the standard group Administrators.

   **Note:** Verify the spelling of the name of the group. The group Administrator does not exist on Windows. The correct name is Administrators.

   d. Enter the user ID and password of the database instance owner for the program.
   e. Click **Next** to proceed.

![Database Loader](image)

*Figure 5-17  Providing the database system information to the installation program*
8. In the next window (Figure 5-18), you provide location information to store the data. As explained before, the WebSphere BI Connect tables are grouped into several categories that are mapped to tablespaces. For optimal performance, allocate directories of different disks for the main table spaces. The default values presented are fine for an initial setup of the product.

**Note:** The database loader program can be used so that the generated SQL statements are stored in a file and not executed. This enables you to review them with your database administrator and execute the SQL scripts manually.

Click **Next**.
9. In the next window, provide the name of the WebSphere BI Connect group, as well as the user IDs and passwords. The values shown in Figure 5-19 are the default values and match the values used in 5.4.1, “Adding user IDs and a group” on page 66. These user IDs are used to access the database.

Click **Next** to proceed.

![Figure 5-19 Providing the user ID, password and group information](image)
10. In the next window (Figure 5-20), provide the name of the location where the common data is stored. *Common data* is the data exchanged between the three different run-time components of the server. In an environment with multiple servers, this should be the name of the mount point on these servers. Assume that each of the three server components runs on a dedicated machine, and a network share must exist that allows the server components to pass information to each other.

Provide the name of this network share as it is known on the server machines and not necessarily on the database server.

**Note:** The value used in Figure 5-20 implies that you will install WebSphere BI Connect on the C: drive of the wbichub machine.

Click **Next**.

![Image of the installation window](image.png)

*Figure 5-20  Naming the mount point where the common data is stored*

11. An installation summary window opens. Click **Next** to start the installation.
12. When all files are copied, you choose whether you want the database loader to run the generated SQL scripts or whether you want to run them manually, for example, after a detailed review of the scripts. See Figure 5-21.

Detailed steps on how to run the scripts manually are provided in a text file Instructions.txt that is stored in the C:\WBIC\Connect\DBLoader\scripts\DB2 folder. This text file also contains information about the process to delete the database, if you need to restart the process due to a failure, for example.

![Database Loader](image)

*Figure 5-21  Choosing to run the scripts now or later*

### 5.4.4 Local validation

Before proceeding, we recommend that you perform some validation. The database loader program creates many objects and the log file is quite large. The following steps perform some validation, but it is not conclusive. You *must* review the log file.

A quick validation can consist of the following actions:

1. Connect to the database with each user ID.

2. Count the stored procedures that were created. The creation of the stored procedures is the last step in the database loader program. As such, if all stored procedures are created, you can conclude that all previous steps were successful and that all objects used by the stored procedures are also created.
Figure 5-22 shows the commands to perform this in a DB2 command window. The number of stored procedures that should be created is 503.

```sql
db2 => connect to bcgapps user bcgcon using <password>

Database Connection Information

Database server = DB2/NT 8.1.4
SQL authorization ID = BCGCON
Local database alias = BCGAPPS

db2 => select count(*) from syscat.procedures where procschema='DB2ADMIN'

1
----------
  503

1 record(s) selected.

db2 => connect reset
DB20000I The SQL command completed successfully.
```

**Figure 5-22** Simple validation process

5.5 **Installing the software for the hub machine**

The previous section discusses the installation and configuration on the machine wbicdata, which is used as the MQ and database server. This section explains how to install and configure WebSphere BI Connect on the machine wbichub. You perform the following tasks:

1. Add a user and group.
2. Install WebSphere BI Connect Advanced.

At the end of this section, you perform a simple validation to make sure that WebSphere BI Connect is operating correctly before you perform any customization.

5.5.1 **Adding a user and group**

An additional user ID is required. The default name is bcguser, which is also a member of the group bcggroup. Since our environment does not have a common security system, such as a Windows domain, we create this user ID and the group on the machine wbichub that will host the WebSphere BI Connect run time.
Use the same steps as explained in 5.4.1, “Adding user IDs and a group” on page 66, to define the user and the group. Be sure to deselect the **User must change password at next logon** option and select the **Password never expires** option.

### 5.5.2 Installing the product code

To install the product code, follow these steps:

1. Launch the main installation program from the product CD. This time select the **Install the Product** option, as shown in Figure 5-23.

![Figure 5-23 Launching the product installation](image)

   - Product Overview
   - ReadMe File
   - InfoCenter Documentation
   - Create the Database
     - Install the Product
   - Exit

   © The IBM logo and WebSphere are trademarks or registered trademarks of IBM Corp. in the US, other countries, or both. All rights reserved 2001-2004.

2. In the Welcome window, click **Next**.
3. Accept the license agreement, and click **Next**.
4. In the next window, provide the name of an installation folder, for example C:\WBICconnect. Click **Next**.
5. Specify which components are going to run on this system (Figure 5-24). Since we intend to create a single server, we select all components. Click Next.

![Figure 5-24 Selecting the components to install]

6. Select the database product, either IBM DB2 or Oracle.
7. Based on your selection, provide the DB2 or Oracle specific information. For DB2, WebSphere BI Connect needs to know the following information, as shown in Figure 5-25:

- The host name of the database server: `wbicdata`
- The port number used by DB2, which is usually `50000`
- The owner name and password: `db2admin`
- The database name: `bcgapps`
- The schema name: `db2admin`

Click **Next** to continue.

![Figure 5-25 Providing database server information](image-url)
8. The installation program connects to the database server and retrieves information that the database loader program created. Figure 5-26 shows the result of these queries. Click **Next** to proceed.

![Figure 5-26 Retrieving information from the database server](image)

The following information has been identified by connecting to the database:

**JDBC Driver Information:**
- Driver Name: IBM DB2 JDBC Universal Driver Architecture
- Driver Version: 1.1.67

**Database Information:**
- Product Name: DB2/NT
- Product Version: GOL00014
- Table count = 164
- View count = 35
- Function count = 35
- Procedure count = 503

**WebSphere Business Integration - Connect Information:**
- Schema Version = 4.2.2.0.174
- Group Name = bggroup
- Console Logon = bgcgroup
- Receiver Logon = bgcgroup
- Document Manager Logon = bgcgroup
- Document Manager mount point URL = C:\Web\Connect\common
9. Now that the installation program has verified that WebSphere BI Connect can access the database, the installation program continues to request information about WebSphere MQ (Figure 5-27). The installation program needs:

- The host name of the MQ server: wbicdata
- The name of the queue manager: partner_e.bcg.queue.manager
- The port number used by the MQ listener: 9999

**Note:** The installation program does not ask for the name of the server connection channel, which is another piece of information that is required to create an MQ client connection channel. WebSphere BI Connect uses a fixed name for this MQ object, java.channel. This object is listed in the MQ definition script provided by WebSphere BI Connect. It was created as part of the MQ setup in 5.4.2, “Configuring WebSphere MQ” on page 68.

Click **Next**.

![Figure 5-27 Providing MQ connection information](image)

10. Indicate that the installation program should create Windows services for the three run-time components. Click **Next**.
11. The installation program asks for configuration parameters for each of the three components. Figure 5-28 shows these parameters for the Community Console component. A similar step follows for the Receiver and Document Manager components. The parameters are:

- Name of the user ID that the component should use to access the database: bcgcon, bcgrecv, or bcgdoc
- Password for that user ID
- HTTP port to be used by the application server for this component
- HTTPS port to be used by the application server for this component

Click **Next**.

![Figure 5-28 Providing information for the Community Console component](image)

12. In the next two windows, you are prompted about the use of RosettaNet and SMTP, which are not used at this time.

13. Finally, you see a summary window. Click **Next** to start the actual installation.
5.5.3 Local validation

The simplest validation that you can perform is to start the components and verify the contents of the log files. To start WebSphere BI Connect, you can either use the Windows Services application (Figure 5-29), or start the application servers from the command line.

![Figure 5-29   Starting WebSphere BI Connect via Windows Services](image-url)
You enter the following command, on a command line, for each component that is executed in the bin directory of that component, as shown in Figure 5-30.

```
startserver server1
```

```
C:\WBIC\console\was\bin\startserver server1
ADMU0116I: Tool information is being logged in file
    C:\WBIC\console\was\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1400

C:\WBIC\console\was\bin>cd C:\WBIC\receiver\was\bin
C:\WBIC\receiver\was\bin\startserver server1
ADMU0116I: Tool information is being logged in file
    C:\WBIC\receiver\was\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1484

C:\WBIC\receiver\was\bin>cd C:\WBIC\router\was\bin
C:\WBIC\router\was\bin\startserver server1
ADMU0116I: Tool information is being logged in file
    C:\WBIC\router\was\logs\server1\startServer.log
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1636
```

Figure 5-30  Starting WebSphere BI Connect from a command line

While the output in Figure 5-30 already gives an indication that the startup was fine, we recommend that you verify the actual log files. Some errors are not sufficient to cause a failure of the startup of the application server, but will result in error situations later when using WebSphere BI Connect.

The log files are called SystemOut.txt, while the error files are called SystemErr.txt. For each component, such a file is created and can be found in C:\WBIC\component name\was\logs\server1, where component name is one of the following names:

- console
- receiver
- router
5.6 Initial configuration of the WebSphere BI Connect server

The main interface of WebSphere BI Connect is browser-based. Log on to WebSphere BI Connect. Use your favorite browser and enter the following address in the address bar:

http://wbichub:58080/console

The console component of WebSphere BI Connect in our environment runs on the server with host name wbichub. The port number 58080 was configured during the installation. Refer to Figure 5-28 on page 86.

A default company called Operator is set up. One user ID hubadmin belongs to this company, and its expired password is Pa55word. Enter this information in the logon section in Figure 5-31. Then click Login. You can consider the company Operator and users belonging to this company as the system owners of a WebSphere BI Connect installation.

Figure 5-31   Welcome window for WebSphere BI Connect
Since the password is expired, you are forced to change it immediately. Enter a new password and click **Save**, as shown in Figure 5-32.

![WebSphere Business Integration Connect Community Console](image)

**Figure 5-32  Changing the password during initial logon**

You see the main interface of WebSphere BI Connect. It consists of a three-level menu bar at the top. The bold items on each level of the menu tell the user where they are currently working. The menu options depend on the profile of the current users. Chapter 7, “Creating a basic B2B exchange” on page 121, explains how to add other users who have fewer options available in the menu.

Under the menu, you see information for regional settings, such as language and time zone, as well as the name of the current user (Figure 5-33).

![WebSphere Business Integration Connect Community Console](image)

**Figure 5-33  Main window of WebSphere BI Connect**

We explore and use most of the menu options in later chapters in this redbook. The top-level menu option Account Admin has to do with managing accounts and profile information. Profiles exist for participants, gateways and B2B capabilities. It is also possible, via this menu option, to add more users or groups to the
current company. These additional users can be considered as additional system-level operators. The manager user of a community participant should be the person who defines application-level users or business-level users.

If you select the top-menu option Viewers, you see the choice to view information about the system at several levels. The Event Viewer allows you to look for detailed information about ongoing events in the system. These events can be related to ongoing or past processing of documents or to actual system-related events. A higher-level viewer is provided for AS2 and RosettaNet. Here, you can search for information specific to AS2 exchanges. We use these tools in later chapters to verify, for example, that a message disposition notification (MDN) has been received for a given document.

The Document Viewer, for which the search options are shown in Figure 5-34, allows the user to find documents independent of how these documents are exchanged. It is a great way for business-level people to determine the status of a document exchanged on a given date or time between two participants.
The Hub Admin menu option enables you to control system-wide settings and features. Under the submenu Hub Configuration, for example, you can find the option to manage XML formats and targets of document flows and document flow definitions themselves.

The submenu Console Configuration contains a few menu options that an administrator should review first before he or she performs any other configuration work. At this level, the hubadmin user can change locale information, permissions, and the password policy. Password policy settings are
shown in Figure 5-35. To make changes to the password policy, click the update icon marked in Figure 5-35. After you enter the changes, click Save.

![WebSphere Business Integration Connect Community Console](image)

**Figure 5-35  Reviewing the password policy**

More features of the product are demonstrated in later chapters. Chapter 6, “Implementing WebSphere BI Connect Advanced for AIX” on page 95, takes you through the implementation of WebSphere BI Connect Advanced on AIX.
Implementing WebSphere BI Connect Advanced for AIX

This chapter explains how to implement WebSphere BI Connect Advanced for AIX. It follows the same structure as Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57.
6.1 Implementation overview

In this chapter, all components, including the database server and the MQ server, are installed on the same machine.

6.2 Verifying software levels on the AIX machine

As for any implementation of a software product, you need to understand the prerequisites for both software and hardware. You must also know how to verify that prerequisites are met.

WebSphere BI Connect relies on services and features of two other products:

- A database manager, such as DB2, which we use in our environment
- WebSphere MQ

The following instructions help you to validate that these products are installed at the correct level and that required features are available.

6.2.1 Verifying DB2

WebSphere BI Connect requires version 8.1 of DB2 and the installation of FixPak 2. To verify the version and whether FixPak is installed, open a command shell and execute the `db2level` program as the database instance owner. The output of this program (see Figure 6-1), tells you which version of DB2 is installed as well as the level of FixPak that is installed.

```bash
# su - db2inst1
$ db2level
DB21085I Instance "db2inst1" uses "32" bits and DB2 code release "SQL08012" with level identifier "02030106". Informational tokens are "DB2 v8.1.1.16", "s030508", "U486566", and FixPak "2".
Product is installed at "/usr/opt/db2_08_01".
```

Figure 6-1 Output of the DB2 command db2level

DB2 can be installed in many different ways. It consists of several components that you can choose to select during the installation. Since WebSphere BI Connect uses a number of stored procedures that are built during the installation of the product, make sure that the DB2 Application Development Toolkit is installed.
To verify if the toolkit is installed, query the database of installed products using the `lslpp -l` command, as shown in Figure 6-2. The package `db2_08-01.adt.rte` is the required component.

```
# lslpp -l | grep db2
  db2_08_01.adt.rte         8.1.1.16  COMMITTED  Base Application Development
  db2_08_01.adt.samples     8.1.1.16  COMMITTED  ADT Sample Programs
  db2_08_01.ca             8.1.1.16  COMMITTED  Configuration Assistant
  db2_08_01.cc             8.1.1.16  COMMITTED  Control Center
  db2_08_01.ch.en_US.iso88591
  db2_08_01.cj             8.1.1.16  COMMITTED  Java Common files
  db2_08_01.client         8.1.1.16  COMMITTED  Base Client Support
  db2_08_01.cnvucs         8.1.1.0   COMMITTED  Code Page Conversion Tables -
  db2_08_01.conn           8.1.1.16  COMMITTED  Connect Support
  db2_08_01.conv           8.1.1.16  COMMITTED  Code Page Conversion Tables
  db2_08_01.cs.rte          8.1.1.16  COMMITTED  Communication Support - TCP/IP
  db2_08_01.das            8.1.1.16  COMMITTED  Administration Server
  db2_08_01.db2.engn       8.1.1.16  COMMITTED  Base DB2 UDB Support
  db2_08_01.db2.rte        8.1.1.16  COMMITTED  Run-time Environment
  db2_08_01.db2.samples    8.1.1.16  COMMITTED  Sample Database Source
  db2_08_01.dc             8.1.1.16  COMMITTED  Development Center
  db2_08_01.dj             8.1.1.16  COMMITTED  DB2 Data Source Support
  db2_08_01.essg           8.1.1.16  COMMITTED  Product Signature for DB2 UDB
  db2_08_01.fs             8.1.1.16  COMMITTED  First Steps
  db2_08_01.icuc           8.1.1.16  COMMITTED  ICU Collation
  db2_08_01.icut           8.1.1.16  COMMITTED  ICU Utilities
  db2_08_01.inst           8.1.1.16  COMMITTED  DB2 Instance Setup Wizard
  db2_08_01.jdbc           8.1.1.16  COMMITTED  Java Support
  db2_08_01.jhlp.en_US.iso88591
  db2_08_01.ldap           8.1.1.16  COMMITTED  DB2 LDAP Support
  db2_08_01.msg.en_US.iso88591
  db2_08_01.pext           8.1.1.16  COMMITTED  Parallel Extension
  db2_08_01.repl           8.1.1.16  COMMITTED  Replication
  db2_08_01.sqlproc        8.1.1.16  COMMITTED  SQL Procedures
```

Figure 6-2  List of installed components for DB2

If this toolkit is not available, add it by running the DB2 installation program one more time. After you add the required components, you must also re-install the FixPak. Then, make sure that the database instance update program is executed. This update is required to make sure that the instance uses the correct level of code.
Figure 6-3 shows the required steps to update the default instance and the administrative instance. These steps are executed as root user when both instances are stopped.

```
# pwd
/usr/opt/db2_08_01/instance

# ./db2iupdt db2inst1
DBI1070I Program db2iupdt completed successfully.

# ./dasupdt
SQL4410W The DB2 Administration Server is not active.
SQL4406W The DB2 Administration Server was started successfully.
DBI1070I Program dasupdt completed successfully.
```

Figure 6-3  Updating the database instance after installing FixPak

### 6.2.2 Verifying WebSphere MQ

To verify the version of WebSphere MQ, you can run the `mqver` program while logged on as root or mqm user. The output of this program is shown in Figure 6-4. CSD04 is required for WebSphere BI Connect. Figure 6-4 shows that we used CSD07. To obtain the required CSD, visit the following Web site, which contains information about the latest CSDs:


```
# mqver
Name: WebSphere MQ
Version: 530.7  CSD07
CMVC level: p530-07-L040527
BuildType: IKAP - (Production)
```

Figure 6-4  Verifying the level of WebSphere MQ

WebSphere BI Connect also uses Java Message Service (JMS), which may be installed on a provided computer. To verify if JMS support is installed, enter the `lslpp -l` command as shown in Figure 6-5. The required component is called `mqm.java.rte`.

```
# pwd
/usr/opt/db2_08_01/instance

# ./db2iupdt db2inst1
DBI1070I Program db2iupdt completed successfully.

# ./dasupdt
SQL4410W The DB2 Administration Server is not active.
SQL4406W The DB2 Administration Server was started successfully.
DBI1070I Program dasupdt completed successfully.
```
Figure 6-5  Verifying the installed components of WebSphere MQ

An additional WebSphere MQ component is required, MQ Publish/Subscribe. This feature is available as a SupportPac and carries the identifier MA0C. To verify if a system has this feature installed, list the contents of the /usr/lpp/mqm/bin directory and search for the modules strmqbrk, endmqbrk, and dspmqbrk, as shown in Figure 6-6.

Note: The lslpp command is of no help here. The installation of SupportPac MA0C consists of unpacking the downloaded tar file into the bin directory of MQ. As such, there is no trace of this component in the database of installed products.

CSD08 of WebSphere MQ installs this component by default, which removes the need to download and install the SupportPac MA0C.
6.3 Enabling DB2 stored procedures

As indicated earlier, WebSphere BI Connect uses several stored procedures to manage its data. These stored procedures are built during the installation of the product.

While a stored procedure is written in SQL, it results in the creation of a loadable module that gets compiled by DB2. This means that a C/C++ compiler must be available on the database server. It must also be configured for use by the user ID that runs the database instance, for example db2inst1.

Here are a couple of solutions that exist for making a C/C++ compiler available to DB2:

- Purchase VisualAge® C++ from IBM. The required version for DB2 is IBM VisualAge C++ Version 5.0.2.3.
- Download and install the free GNU C/C++ compiler.

To verify if the compiler is installed and the version it has, use the `lslpp -l` command, as shown in Figure 6-7.

```
# lslpp -l | grep vacpp.cmp
  vacpp.cmp.aix50.lib    6.0.0.0 COMMITTED VisualAge C++ Libraries for
  vacpp.cmp.aix50.tools   6.0.0.0 COMMITTED VisualAge C++ Tools for AIX
  vacpp.cmp.core         6.0.0.0 COMMITTED VisualAge C++ Compiler
  vacpp.cmp.include      6.0.0.0 COMMITTED VisualAge C++ Compiler Incl
  vacpp.cmp.lib          6.0.0.0 COMMITTED VisualAge C++ Libraries
  vacpp.cmp.rte          6.0.0.0 COMMITTED VisualAge C++ Compiler
  vacpp.cmp.tools        6.0.0.0 COMMITTED VisualAge C++ Tools
  vacpp.cmp.core         6.0.0.0 COMMITTED VisualAge C++ Compiler
```

*Figure 6-7 Verifying the availability of a C++ compiler*

DB2 provides a compilation script that works with VisualAge C++. You only need to verify the script sr_cpath, which is in `/home/db2inst1/sqlib/function/routine`. This script adds a number of directories to the PATH environment variable. As long as the C++ compiler is installed in the directory added to the PATH environment variable, you do not need to do anything special.
For other compilers or to have a customized compilation procedure, you need to use two DB2 environment variables:

- **DB2_SQLROUTINE_COMPILER_PATH**
  
  This variable is assigned the path name of a script that sets the compiler environment variables.

- **DB2_SQLROUTINE_COMPILE_COMMAND**
  
  This variable is assigned the full command that DB2 uses to compile the C files generated for SQL procedures.

You can find full details about the setup in the DB2 Information Management Software Information Center at:

http://publib.boulder.ibm.com/infocenter/db2help/index.jsp

When you reach the Information Center, in the menu on the left, select Tasks → Developing applications → Setting up the application development environment → SQL procedures.

### 6.4 Software installation

This section explains how to install and configure WebSphere BI Connect. This includes:

- Adding users and groups
- Configuring WebSphere MQ
- Installing the database loader and creating the database
- Installing WebSphere BI Connect Advanced itself

At the end of this section, before you do any customization, you perform a simple validation to ensure that WebSphere BI Connect is operating correctly.
6.4.1 Adding user accounts

WebSphere BI Connect uses four different user IDs and one group to manage security roles. The names of these user IDs and group are free to choose. However, we recommend that you use the default names.

To define user IDs and groups, use the system management utility smitty, or use the following commands.

To add a group bcguser, use:

```
mkgroup -A bcggroup
```

To add a user bcguser that has bcggroup has its primary group, use:

```
mkuser pgrp=bcggroup bcguser
```

To set the initial expired password, use:

```
passwd bcguser
```

Be aware that this sets only a temporary password. To change it to a permanent value, log on via Telnet or another way as bcguser.

In the same way, define user IDs for the roles bcgcon, bcgrecv, and bcgdoc.

6.4.2 Configuring WebSphere MQ

You must create and configure several MQ resources for use by WebSphere BI - Connect. First, you must create a queue manager.

When creating a queue manager, you can customize the size of the MQ transaction log. Two types of logs exist, circular and linear. A circular log provides a transactional log, but is easier to manage. A linear log is required to make and restore backups within WebSphere MQ and to recover from hard disk failures, assuming that messages and log are not stored on the same disk.

The logging can be spread over multiple log files, and the size of the log file can also be configured. To make the log a little bigger than the default size, set the log file size to 1024, which is 1024 pages of 4 KB.

The command to create the queue manager needs to be executed as mqm user and is shown in Figure 6-8. The ctmqm command is followed by the strmqm command to start the queue manager. The setmqcap command is required to set the number of purchased licenses for the product WebSphere MQ. The number of required licenses depends on the number of processors in the system.
Next, you must enter two more commands to configure the MQ resources. The first command (see Figure 6-9) refers to a definition file that is installed as part of the Java Messaging feature of WebSphere MQ. It contains MQ definitions used by JMS Publish/Subscribe. The second command defines the number of MQ resources used by WebSphere BI Connect. It is available on the product CD, mounted on /cdrom (see Figure 6-9).

```
$ runmqsc partner_a.bcg.queue.manager < /usr/lpp/mqm/java/bin/MQJMS_PSQ.mqsc
$ runmqsc partner_a.bcg.queue.manager < /cdrom/Tools/MQSeries/create_wbic_queues.mqsc
```

Finally, you must make a few more changes to the run-time parameters of the queue manager. In a text editor, open the qm.ini file, which is located in /var/mqm/qmgrs/partner_a.bcg.queue.manager. Add the stanza Channels at the end of the text file. Underneath these, add the following parameters:

- MaxChannels
- MaxActiveChannels
Refer to Figure 6-10. Make sure that the file ends with an empty line.

![Figure 6-10](image)

This change ensures that WebSphere BI Connect can establish sufficient MQ client sessions when required.
6.4.3 Installing the database schema

In addition to WebSphere MQ, WebSphere BI Connect relies on the services of a database manager. For this redbook, we used DB2. However, other database products are also supported. The database is used to store data that can be classified in two broad categories: profile information and logging information.

The profile information is largely a read-only category when the system is configured. Changes occur only when new profiles are added or when an existing profile is updated, for example, replacing an expired certificate.

The logging information is a highly active category of information. A single document exchange results in many logging events, which is useful in keeping track of what happens in the system or what has happened in the system. Thus, this category of information expands constantly and is consulted mostly by online users and administrators.

Given the different nature of the database access to both categories of information, it comes as no surprise that WebSphere BI Connect has grouped them in two different database tablespaces so that it becomes easy to manage them separately. This can mean storing the information on separate disks and implementing different backup and reorganization procedures.

1. Start the product installation program from the product CD.
2. You are presented with several options, among which the two main tasks are:
   - Create the Database
   - Install the Product

Select the Create the Database option to launch the database loader program (see Figure 6-11). You must run this installation program as a root user. However, the database is created with the authority of the database instance owner, which is information that you need to provide during the installation.
3. When the database loader program is started, the welcome window opens. Click **Next**.

4. In the next window, accept the software license. Click **Next**.

5. Provide an installation directory (for example, `/opt/IBM/WBIConnect/DBLoader`). Click **Next**.

6. Select the database server product that is going to be used. This can be:
   - IBM DB2 8.1.2 or later
   - Oracle 9i 9.2.0 or later

   The following figures show the choices that we made for our environment and are specific for DB2. The product documentation explains this for Oracle as well.
7. In the Database Loader window (Figure 6-12) that opens, enter the following information:
   a. Enter the name of the new database for use by WebSphere BI Connect, which by default is BCGAPPS.
   b. Type the name of the database instance. A standard UNIX installation of DB2 has a default instance called db2inst1. Other instances may also be created. This instance must exist before you run the database loader program.
   c. Specify the name of the group to which the database owner user ID belongs. On UNIX, this is usually the standard group db2grp1.
   d. Enter user ID and password of the database instance owner.
   e. Click Next to proceed.

Figure 6-12 Providing database system information to the installation program
8. In the next window (Figure 6-13), provide the location information to store the data. As explained earlier, the WebSphere BI Connect tables are grouped in categories that are mapped to table spaces. For optimal performance, allocate directories of different disks for the main table spaces. The presented default values are fine for an initial setup of the product. Click **Next**.

**Note:** You can use the database loader program so that the generated SQL statements are stored in a file and not executed. That way, you can review them with your database administrator and execute the SQL scripts manually.

![Database Loader](image)

*Figure 6-13 Providing location information for the table spaces*
9. Provide the name of the WebSphere BI Connect group and user IDs and their passwords (Figure 6-14). The values shown are the default values and match the values used in 6.4.1, “Adding user accounts” on page 102. These user IDs are used to access the database.

Click **Next** to proceed.

*Figure 6-14 Providing user ID, password and group information*
10. Name the mount point for the common data, which by default is 
/opt/IBM/WBICConnect/common.

Figure 6-15 shows a warning message that you may encounter at this time. In 
this scenario, we rely on the script sr_cpath to set the environment variable 
PATH for DB2 during the process of creating stored procedures. Therefore, 
the current value of the environment variable PATH does not contain the 
directory that contains the C compiler. We recommend that you always review 
your setup when you see this warning. In our environment, we can ignore this 
warning.

Click **Next**.

![Figure 6-15   Warning about the C compiler](image)

11. Review the installation summary and click **Next** to start the installation.
12. When all files are copied, you can choose whether you want the database loader to execute the generated SQL scripts or whether you want to run them manually, for example, after a detailed review of the scripts (see Figure 6-16). You can find detailed steps about how to run the scripts manually in the Instructions.txt file, stored in the /opt/IBM/WBICConnect/DBLoader/scripts/DB2 folder. This text file also contains information about the process to delete the database, if you need to restart the process due to a failure, for example. Click Next.

![Database Loader dialog box](image)

Figure 6-16  Choosing whether to run the scripts now or later

Before proceeding, we recommend that you validate the installation. The database loader program creates many objects, and the log files that are available in /tmp/WBICConnect/logs are quite large. The following steps explain how to perform some validation. However, it is not conclusive. You may still need to review the log files.
A quick validation can consist of the following tasks:

1. Connect to the database with each user ID.
2. Count the stored procedures that were created.

Figure 6-17 shows the commands to perform this in a DB2 command session, executed while logged on as db2inst1. The number of stored procedures that should be created is 503.

```
db2 => connect to bcgapps user bcgcon using byte2eat

   Database Connection Information
   Database server        = DB2/6000 8.1.2
   SQL authorization ID   = BCGCON
   Local database alias   = BCGAPPS

   db2 => select count(*) from syscat.procedures where procschema='DB2INST1'
   1
   -----------
   503
   1 record(s) selected.

   db2 => connect reset
   DB20000I The SQL command completed successfully.
```

*Figure 6-17  Simple validation process*
6.4.4 Installing the product code

After you install the database schema, follow these steps.

1. The launchpad should still be active. If not, restart it from the product CD. This time, select **Install the Product** (see Figure 6-18).

![Figure 6-18 Launching the product installation](image)

2. In the welcome window, click **Next**.
3. Accept the license and click **Next**.
4. Provide the name of an installation folder, for example `/opt/IBM/WBICconnect`. 
5. In the next window (Figure 6-19), specify which components are going to run on this system. Since we are creating a single server, we select all components. Click Next.

6. Select the database product, either IBM DB2 or Oracle.
7. Based on your selection in the previous step, you are prompted for specific information about your selection. In this example, for DB2, WebSphere BI Connect needs to know the following information, as shown in Figure 6-20:

- The host name of the server
- The port number used by DB2, which is usually 50000
- The owner name and password: db2inst1
- The database name: bcgapps
- The schema name: db2inst1

Click **Next** to continue.

![Figure 6-20  Providing the database server information](image)
8. The installation program now connects to the database server and retrieves information that the database loader program was created. Figure 6-21 shows the result of these queries. Click **Next** to proceed.
9. The installation program now asks for the user ID and password of the user who will own the product files and executes the processes, which is bcguser (see Figure 6-22). Click **Next** to proceed.

![Figure 6-22 Providing user information](image)

*Figure 6-22 Providing user information*
10. Now that the installation program has verified that WebSphere BI Connect can access the database, the installation program continues to request information about WebSphere MQ (see Figure 6-23). The installation program needs:

- The host name of the MQ server
- The name of the queue manager, partner_a.bc.g.queue.manager
- The port number used by the MQ listener

**Note:** The installation program does not ask for the name of the server connection channel, which is another piece of information that is required to create an MQ client connection channel. WebSphere BI Connect uses a fixed name for this MQ object, java.channel. This object is listed in the MQ definition script provided by WebSphere BI Connect. It was created as part of the MQ setup in 6.4.2, “Configuring WebSphere MQ” on page 102.
11. The next window (Figure 6-24) prompts you for the configuration parameters for each of the three components. In Figure 6-24, you see these parameters for the Community Console component. A similar step follows for the Receiver and Document Manager components. The parameters are:

- Name of the user ID that the component should use to access the database, which is bcgcon, bcgrecv, or bcgdoc
- Password for that user ID
- HTTP port to be used by the application server for this component
- HTTPS port to be used by the application server for this component

Click **Next**.

![Figure 6-24 Providing information for the Community Console component](image)

12. In the next two windows, you are prompted about the use of RosettaNet and SMTP, which are not used at this time.

13. Finally, you see a summary window. Click **Next** to start the actual installation.
6.4.5 Local validation

The simplest validation that you can perform is to start the components and verify the contents of the log files. To start WebSphere BI Connect, start the application servers from a command line.

When you start WebSphere BI Connect, you start three different application servers. Since this can require a lot of typing, you can create a simple script, such as the one shown in Figure 6-25, to start all three servers.

![Simple script file to launch all components of WebSphere BI Connect](image)

When starting WebSphere BI Connect for the first time, it is a good idea to verify the actual log files. Some errors are not sufficient to cause a failure of the startup of the application server, but result in error situations later when using WebSphere BI Connect.

The log files are called SystemOut.txt, while the error files are called SystemErr.txt. For each component such a file is created. You can find it in /opt/IBM/WBICOnnect/component name/was/logs/server1, where component name is one of the following names:

- console
- receiver
- router

6.5 Initial configuration of the WebSphere BI Connect server

When the server is running, the difference between an AIX instance of WebSphere BI Connect and a Windows instance of WebSphere BI Connect becomes small. The interface is always browser based. Therefore, in this chapter, we do not repeat the initial logon process on AIX. Refer to 5.6, “Initial configuration of the WebSphere BI Connect server” on page 89, for more information.
Creating a basic B2B exchange

This chapter describes the required setup to enable AS2 communication between the WebSphere BI Connect installations created in Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57, and in Chapter 6, “Implementing WebSphere BI Connect Advanced for AIX” on page 95.

In this chapter, we limit the communication to the exchange of electronic data interchange (EDI) documents retrieved from the file system and stored on the file system. Later chapters discuss the use of message queues as the source and target of the document exchange between both partners.
7.1 Scenario overview

This scenario entails working with the two WebSphere BI Connect instances that were implemented in the previous chapters. The first instance is running WebSphere BI Connect Advanced on AIX. The second instance is running WebSphere BI Connect Enterprise on Windows 2000.

This chapter explains how to implement the exchange of AS2-EDI documents between both partners (Figure 7-1). For both partners, the source and target of the EDI documents are the file system.

![Figure 7-1 Scenario overview](image)

7.1.1 Role-based configuration

When working with WebSphere BI Connect, a hub administrator needs to set up profiles for participants. A participant consists of a collection of one or more users, and one user (admin) is always defined. Thus, when implementing WebSphere BI Connect, you always have at least three company profiles:

- The Operator (built-in) company can be seen as the IT group that operates the system.
- A participant, who acts as community manager, can be considered as the business owner of the system.
- A participant, who acts as community participant, may be a community manager of their own WebSphere BI Connect system and may be the business owner of their own WebSphere BI Connect system.
Each company profile contains at least one user ID, which is admin for participants and hubadmin for the company Operator.

When participants are created, the three users can all participate in setting up the configuration of a given system. Each user has a set of authorities to perform certain tasks. Alternatively, the hubadmin fully defines the profiles of the two community participants. The first approach promotes self-management and collaboration between the different users. The latter approach, where the hubadmin takes full control, can be used when enterprises prefer to not share their environment and to keep full control of their system.

Both approaches are equally valid. To demonstrate the concept of participating partners in a B2B community, this chapter describes a setup where each of three players has an active role. It briefly discusses how the alternative implementation would have been performed.

### 7.1.2 Outbound flow

For the outbound flow, an EDI document is stored for delivery by an internal application in a dedicated folder on the file system. The arrival of this file is detected by the Receiver component of WebSphere BI Connect. The document is passed on to the Document Manager component of WebSphere BI Connect.

The Document Manager parses the EDI document to find the source and target business identifier in the document. These identifiers, which are part of the ISA segment of an EDI document, need to be defined within WebSphere BI Connect.

Given the type of document, the source and target partners, the Document Manager tries to find a participant connection that matches these parameters. This means that you need to tell WebSphere BI Connect that these business partners support EDI documents. Or, speaking in WebSphere BI Connect terms, the Document Manager will review the B2B capabilities of the two involved partners. Thus, you need to provide the B2B capabilities of both partners to the WebSphere BI Connect server.

Assuming that the system finds a match, the Document Manager looks for a document flow that details what needs to be done to send an EDI document from the source partner to the target partner. This is called an *interaction*. In this interaction, you describe how an EDI document is packaged as an AS2 document.

Finally, the Document Manager needs to know to where the document needs to be sent. This information is called the *gateway*. The target partner needs to provide a gateway destination, which is an HTTP gateway in this scenario.
Three different players must provide the information that WebSphere BI Connect needs to perform this outbound flow (Figure 7-2):

- The hubadmin who is responsible for the correct operation of the WebSphere BI Connect server. This person needs to perform the following tasks:
  - Define the location where the WebSphere BI Connect server will look for the EDI documents that need to be sent (integration target).
  - Define the interaction to package EDI documents in an AS2 format (document flow definition).
  - Define the profile of the community participant who owns the server and provide the business identifier. This participant is the community manager.
  - Define the profile of the community participant who is the target of the exchange and provide the business identifier.
  - Define the connection between the two participants.
- The community manager who owns this installation of WebSphere BI Connect. This person defines their B2B capabilities to send EDI documents.
- The community participant who is the target partner of this outbound flow. This person must:
  - Define their B2B capabilities to receive EDI documents packaged in AS2.
  - Provide the information about their target URI, called a gateway.

![Figure 7-2 Outbound flow](image-url)
7.1.3 Inbound flow

When a business partner sends an EDI document over AS2 to the WebSphere BI Connect server, there needs to be a URL on which the server is listening for incoming HTTP datastreams. This URL is called a target and needs to be defined to the WebSphere BI Connect server.

Upon arrival of an EDI document, the Receiver stores it in the common storage of the server. The Document Manager retrieves it from the common storage and parses the document. It again looks for the business identifiers in that document and locates a connection between the two partners that are named in the EDI document.

If such a connection is found, the WebSphere BI Connect server performs the interaction that was set up for it. This means the AS2 packaging will be removed and the unpackaged EDI document will be passed to the gateway that was set up for this connection.

Three different players must provide the information that WebSphere BI Connect needs to perform this inbound flow (Figure 7-3):

- The hubadmin who is responsible for the correct operation of the WebSphere BI Connect server. This person needs to perform the following tasks:
  - Define the location that can be used to send documents, which is an HTTP target.
  - Define the interaction to unpackage AS2 EDI documents.
  - Define the profile of the community participant who owns the server and provides the business identifier, which was already done for the outbound flow.
  - Define the profile of the community participant who is the target of the exchange and provides the business identifier, which was already done for the outbound flow.
  - Define the connection between the two participants.

- The community participant who owns this installation of WebSphere BI Connect. This person is given the role of community manager, and needs to:
  - Define their B2B capabilities to receive EDI documents.
  - Provide the information where WebSphere BI Connect needs to deliver the EDI document, which is the file system gateway.

- The community participant who is the target partner of this outbound flow. This person must define their B2B capabilities to send AS2 packaged EDI documents.
This chapter describes the steps that each of the three players needs to perform. However, the assumption is that EDI documents are going to be sent in both directions. Thus, in the discussion about the setup performed by a participant, you also learn about the steps to set up dual communication without necessarily making a distinction between the required setup for inbound or outbound communication.

7.2 Configurations tasks for hubadmin of Company E

The hub administrator has the responsibility to look after the overall configuration of the WebSphere BI Connect server. They have the task to create profiles for the community manager and community participants. They also need to look after configuration tasks that are global settings, which are applicable for all community participants (Figure 7-4).
Chapter 7. Creating a basic B2B exchange

7.2.1 Creating targets

Targets are the entry points into the WebSphere BI Connect server. This entry point can be a directory, a queue, an HTTP URI, or some other resource. Targets need to be created for sending documents to partners. Such a target is the location where internal applications drop documents to be sent to partners. Targets need to be created for receiving documents from partners. The value of this target is what you need to provide to your partners so that they can send information to you.

To create a target, follow these steps:

1. Log on as the hub administrator (user ID hubadmin, company Operator). From the WebSphere BI Connect menu, select Hub Admin → Hub Configuration → Targets.
2. The browser shows a list of defined targets (Figure 7-5), which is currently empty. Click **Create Target**.

![Figure 7-5 List of targets in WebSphere BI Connect](image)

3. In the Target Details panel, follow these steps:
   a. Enter a name, and for Transport, select **HTTP/S**.
      
      A target is not really linked to a specific community participant, nor is it linked to a specific type of document. You can use specific names to make it clear why a certain target was created. Throughout this redbook, we make the name explicit to assist you in understanding where an object was defined and for what purposes.

      After you select a transport, the interface displays additional properties specific for that transport.
   b. Specify whether this target will be used for test or production purposes.
c. Since we are creating an HTTP/S target, provide a URI. This URI must start with /bcgreceiver, but can be named freely beyond that.

d. Click **Save** (Figure 7-6).
4. When the new target is created, click **List** to return to the list of targets (Figure 7-7).

![Figure 7-7](image)

5. When you see the list of targets, click **Create Target** again to create a second target, this time for picking up EDI documents to be sent to the partners of Company E.
6. In the Target Details panel (Figure 7-8), complete these tasks:

a. Provide a name for the target.

b. For Transport, select **File Directory**. The name of the directory can again provide an indication of the usage of the directory, but it is not required. Note that WebSphere BI Connect creates the directory edi_out, but the other directories need to be created before the target is created.

c. Optionally, you can provide values to control the polling frequency of this directory. Also, for systems where many files need to be sent, this may be required to increase the number of threads.

d. Click **Save** to store the new target in the database.

![Image](image-url)  
**Figure 7-8** Creating a target of type File Directory

This completes the creation of the object HTTPTarget in Figure 7-3 on page 126 and the object FileSystemTarget in Figure 7-2 on page 124.
7.2.2 Creating interactions

Document flow definitions are constructs that determine the core functionality of WebSphere BI Connect. Several types of document flow exist, such as package flows, protocol flows, or document flows. A package flow is the logic that is required to package a document according to a specific specification, such as AS2. A protocol flow is the logic that is required to process a document that adheres to a certain protocol, such as EDI-X12. A document flow details what needs to be done with the contents of a flow. This can be as simple as Pass Through or executing a transformation map or a concatenation of several actions in a single flow.

To work with document flows, follow these steps:

1. Select Hub Admin → Hub Configuration → Document Flow Definitions (Figure 7-9).

Note: The hub administrator performs this task. As a community manager or participant, you need to define B2B capabilities, but these are in essence a subset of what is enabled by the hub administrator.
2. By default, several packaging methods are enabled, as shown in Figure 7-9. When you click the icon next to a package, you see the protocols that are enabled or disabled for a given packaging. This tells you that the WebSphere BI Connect server is configured to package AS2 and knows about packaging EDI as a protocol into AS2. You must create a sequence of actions that tells the WebSphere BI Connect server how to go from an AS2 packaged EDI document to an unpackaged EDI document. This sequence is an interaction.

Click Manage Interactions to work with interactions.

Figure 7-9  List of document flows
3. In the Manage Interactions panel (Figure 7-10), click Search to see the a number of interactions that exist. In this example, no interaction is predefined for the combination of EDI-X12 and AS2. The top four interactions listed in Figure 7-14 on page 138 are predefined. They cover the AS2 and Backend Integration packaging of binary documents. Backend Integration packaging is basically Java Message Service (JMS) packaging.

Since no interaction for EDI is predefined, click Create Interaction to add it.

![WebSphere Business Integration Connect Community Console](image)

*Figure 7-10 Searching for interactions*
4. Figure 7-11 shows the starting point to create a new interaction. You see the existing packages under the headings Source and Target. Referring to Figure 7-2 on page 124 and Figure 7-3 on page 126, you must indicate the flow for WebSphere BI Connect to go from:

- Unpackaged EDI documents to AS2 packaged EDI documents for the outbound flow
- AS2 packaged EDI documents to unpackaged EDI documents for the inbound flow

Follow these steps:

a. Starting with the inbound flow, under Source, click the icon next to **Package: AS**.

![WebSphere Business Integration Connect Community Console](image)

*Figure 7-11 Creating a new interaction*
b. Now you see several protocols (Figure 7-12), such as EDI-X12. Click the icon next to Protocol: EDI-X12. Then select Document Flow: ALL.

c. Under Target, click the icon next to Package None. Click the icon next to Protocol: EDI-X12 and select Document Flow: ALL.

d. For Action, select Pass Through.

e. Click Save.

Other actions exist as well. Most are related to processing custom XML documents and RosettaNet documents. You can also add more actions. For the scenarios in this chapter, the action Pass Through is sufficient.
5. Using similar steps, create the interaction to go from unpackaged EDI-X12 documents to AS2 packaged EDI documents, as shown in Figure 7-13.

   a. Select the **Pass Through** action.
   b. Click **Save** to save the interaction.
   c. Click **Manage Document Flow Definitions**.

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-13** Unpackaged EDI-X12 to AS2 packaged EDI-X12
You return to the search window (see Figure 7-10 on page 134). This time the search returns two new interactions, at the bottom of the list (see Figure 7-14).

You have now created the two objects labeled *Interactions* in Figure 7-2 on page 124 and Figure 7-3 on page 126.
7.2.3 Creating a community manager

The WebSphere BI Connect server can be used by a number of community participants. One of them can be defined as community manager, which is usually the company that owns the WebSphere BI Connect server.

To create a community manager, follow these steps:

1. While logged on as the hub administrator, select **Account Admin → Profiles → Community Participant**.

2. Click **Create** to add a new participant, as shown in Figure 7-15.

![Creating new participant: Selecting Create](image)
3. To create a new participant, you must (see Figure 7-16):
   a. Provide a participant login name, which is case-sensitive and can’t contain blanks. This name is actually used as company name in the login window.
   b. Provide a participant name, which can be any text.
   c. Provide a participant type, which is either community manager or community participant.
   d. Optionally provide a vendor type, which can have values such as Supplier or Distributor.
   e. Optionally provide a Web site. This value does not refer to the AS2 server, but can refer to the public Web site of that company.

![Figure 7-16 Creating a new participant: Providing information](image)

Within B2B, several techniques exist to provide unique or pseudo-unique identifications. An example of unique identifications are the Data Universal Numbering System (DUNS) numbers. DUNS numbers are assigned by Dun & Bradstreet Corporation. For more information about DUNS numbers, refer to:

http://www.dnb.com/
While DUNS numbers can be unique, other schemas exist that are only pseudo-unique, which means that there is no global organization that manages the assignment of the identifiers. An example of pseudo-unique identifiers are the identifiers used in the ISA segment of an EDI document. These identifiers tend to be unique within a given EDI network of business partners. The EDI network provider makes sure that the identifiers are unique within the scope of their network.

a. To add an EDI identifier to a participant profile, under Business ID, click New.

b. For Type, select Freeform as the type of business ID.

c. For Identifier, specify a value. We used the value companye, which is used in the following situations (Figure 7-17).

- The companye value is going to be recognized as part of the ISA segment of an EDI document. This means that WebSphere BI Connect will parse an EDI document and locate the EDI identifiers to look up participants and participant connections.
- The companye value will be used to create the AS2 package to set the value in the AS2 From field. For incoming AS2 documents, the value for the AS2 From and To fields will be used to retrieve the participants and the participant connections.

**Note:** It is possible to configure WebSphere BI Connect so that the AS2 IDs can be different from the EDI identifiers. You do this by setting the appropriate attributes in the participant connection. See 7.5, “Connecting Company E to Company A” on page 163.
d. Under IP Address or Host Name, click **New**.

e. Provide the name or IP address for the gateway that needs to be used for the community manager.

f. Click **Save** in the middle of the page to store this new profile to the database.

*Figure 7-17  Creating a new participant: Business ID, IP address or host name*
When the new profile is created, WebSphere BI Connect generates a temporary password. This password must be passed to the administrator for the community manager company (Figure 7-18).

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-18  New profile created**

You have now completed the definition of the object labeled From in Figure 7-2 on page 124 and the object labeled To in Figure 7-3 on page 126. These objects are used in the interpretation of EDI documents.
**Note:** If the generated password becomes lost or, in general, if a password is lost, the hubadmin can perform a reset of the password.

1. Log on as hubadmin.
2. Select **Account Admin → Profiles → Community Participant**.
3. In the Participant Search window, click **Search**.
4. You see a list of company profiles. Open the details of the company to which the user belongs that needs a new password.
5. From the menu, select **Users**.
6. You see a list of users who belong to this company profile. Open the details of that user, for example admin.
7. Click the edit record icon.
8. Set the password or generate a new password for this user (Figure 7-19).

---

*Figure 7-19  Changing the password of a user*
7.2.4 Creating a community participant

The same steps to create the community manager are used to create a community participant. The participant here is Company A, which uses WebSphere BI Connect Advanced, and the login name is companyA. Refer to Figure 7-20.

The business identifier companya is going to be used as the value for the object labeled To in Figure 7-2 on page 124 and the object labeled From in Figure 7-3 on page 126. These objects are used in the interpretation of EDI documents.
Before switching roles, let’s summarize the configuration work done so far (see Figure 7-21). For the inbound flow, we have defined:

- The target HTTP Target
- The interaction from AS2 packaged EDI to unpackaged EDI
- The IDs From and To as used in the EDI document and the AS2 packaging

**Note:** The hub administrator may not necessarily have information such as the business identifier from the community participant. However, when the community participant logs on for the first time, they can update their profile and add this information to it.
For the outbound flow (Figure 7-22), we have defined:

- The target FileSystemTarget
- The interaction from unpackaged EDI to AS2 packaged EDI
- The IDs From and To as used in the EDI document and the AS2 packaging

At this time, the hub operator has performed the necessary tasks to provide access to the community manager and the community participants. This makes it possible for them to provide the required information about what they want WebSphere BI Connect to do for them.

Figure 7-22  Status of configuration for outbound flow
7.3 Configuration tasks by Company E administrator

The previous sections explain how to define the community manager and named it Company E. The administrator for Company E now needs to perform his own part of the configuration work (Figure 7-23).

Figure 7-23 Logging on as admin of community manager to server of Company E

7.3.1 Initial logon by community manager

The login name defined in 7.2.3, “Creating a community manager” on page 139, is not the same as the user name. It is the value used in the field Company Name in the Welcome window of the Community Console. For each new company, a predefined user name is created, which is admin.

Using the user name admin, the company name companyE, and the generated password (see Figure 7-18 on page 143), log on to WebSphere BI Connect (Figure 7-24).
Before you see the main window, WebSphere BI Connect requests that the administrator of Company E changes his password.

If required or desired, the company administrator can now add users and groups for other people within Company E that need to interact with the Community Console.

### 7.3.2 Creating a gateway

To proceed with the setup for enabling the exchange of EDI documents over AS2 between Company E and Company A, the administrator of Company E needs to create a gateway. A gateway is like an exit point out of the WebSphere BI Connect server and into the world of Company E. Where does WebSphere BI Connect need to deliver documents intended for Company E?

For the scenario in this chapter, we use a file-based gateway. The assumption is that this gateway should already exist as an entry point into internal applications or systems. Therefore, make sure that the complete directory structure exists before you define the gateway in WebSphere BI Connect.

**Attention:** This is different from the file system target where WebSphere BI Connect creates the directory, assuming that higher-level directories exist. Here, make sure that every level of the path is predefined.
1. Using Explorer, create a directory to store incoming EDI documents. For Company E, we used the value `\WBIConnect\data\companye\edi_in`.

2. Return to the Community Console and select **Account Admin → Profiles → Gateways**.

3. You see an empty list of gateways. Click **Create**.

4. In the Gateway List panel (Figure 7-25), complete these tasks:
   a. Provide a name for the gateway and select the transport, which is File Directory for our scenario.
   b. Enter the complete directory structure in the URI format. In our case, there are three slash (/) characters following the colon character. If a disk letter is required, add `C:` between the second and third slash character. In the URI format, the Windows directory separator back-slash (`\`) is not used.
   c. Click **Save** to store the new gateway in the database.
   d. Click **List** to return to the list of defined gateways.

---

**Figure 7-25 Creating a new gateway**
5. Figure 7-26 lists the new gateway. It also shows that a default gateway has not been provided. To label a gateway as the default gateway, click View Default Gateways.

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-26** List of defined gateways

---

**Note:** Compare the menu options for the hub administrator in Figure 7-15 on page 139 with the menu options for a company administrator in Figure 7-25. The menu option Hub Admin is unavailable for the company administrator, preventing him to create targets and document flow definitions.
6. Figure 7-27 shows how to label a default gateway for each type of gateway. Click **Save** to store your selection. Click **List**.

![Figure 7-27  Label a gateway as the default gateway](image)

---

152  B2B Solutions using WebSphere BI Connect Version 4.2.2
You return to the list of gateways (Figure 7-28)

Referring to Figure 7-21 on page 146, you have now completed the setup for the object labeled FileSystemGateway.
7.3.3 Providing B2B capabilities

Earlier, we configured the interactions on the WebSphere BI Connect server. Within the possible interactions that are defined on a server, a given participant optionally can enable them.

1. While logged on as the administrator of Company E, select **Account Admin → Profiles → B2B Capabilities** (Figure 7-29).

![Figure 7-29 Providing B2B capabilities for Company E](image)
2. As shown in Figure 7-30, complete these tasks:
   a. Click the icon underneath Set Source for Package: AS to enable. Click the icon underneath Set Target for Package: AS to enable.
   b. Click the icon next to Package: AS to drill down. Click the icon for Protocol: EDI-X12 (ALL) for both source and target.
   c. Click the icon next to Protocol: EDI-X12(ALL) to drill down. Click the icon for Document Flow: ALL for both source and target.
   d. Repeat this sequence for Package: None.

At the end, your display should look similar to the example in Figure 7-30.

Figure 7-30 Completed B2B capabilities

The participant says that they have the capability to receive and send EDI documents packaged in AS2 or not packaged. However, the actual action is set by the hub administrator, in our case Pass Through.
If you select the edit icon next to Package: AS, you can set several AS-related parameters, such as encryption or custom AS2 identifiers. These parameters can also be set at the level of a participant connection, so that the values are specific for a single pair of partners. If you set values for the AS2 parameters in Figure 7-30, they become the default values for all AS2 connections that involve Company E.

The configuration work for the administrator of Company E is now complete.

### 7.4 Configuration tasks for partner Company A

Earlier in 7.2.4, “Creating a community participant” on page 145, you created a participant profile for Company A. A representative of Company A can now use the new profile and log on using the generated password to the Web site of their partner Company E. Thus, the hub administrator defines the profile, but the administrator of the partner Company A completes the partner profile.

Figure 7-31 illustrates what happens after you log on as the administrator of the community participant to Company E.
1. Log on to the hub of Company E (see Figure 7-31). Use these values:
   - User name: admin
   - Password: Generated by WebSphere BI Connect and provided by the hub administrator of Company E
   - Company: companyA, as defined in 7.2.4, “Creating a community participant” on page 145

2. After the initial logon, the administrator of Company A again must provide a new password. Similar to what the community manager needed to do, the community participant (Company A) must set their B2B capabilities and create a gateway.

3. Select **Account Admin → Profiles → Gateways**.

4. When you see the empty list of gateways, click **Create**.
5. In the Gateway List panel (Figure 7-32), complete these tasks:
   a. Provide a name for the gateway.
   b. Select the transport, which is HTTP/S.
   c. Provide the target URI.

   **Note:** The target URI needs to match with a target object defined on the WebSphere BI Connect server of Company A. The exit point (gateway) on WebSphere BI Connect for Company E needs to match with the entry point (target) on WebSphere BI Connect for Company A.

d. Click **Save**.

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-32 Creating a gateway for Company A**

The WebSphere BI Connect server of Company A runs on AIX. Its implementation is discussed in Chapter 6, “Implementing WebSphere BI...”
Connect Advanced for AIX” on page 95. Its configuration to enable AS2 EDI documentation is discussed starting with 7.6, “Configuration tasks for hubadmin of Company A” on page 170.

6. When the gateway object is stored in the database, click **List** to return to the list of gateways.

7. The list of gateways does not have a default gateway. Click **View Default Gateways** to label the HTTPGateway object as the default gateway for gateway type Production.

The list of gateways for Company A should now look like the example in Figure 7-33.

![Figure 7-33  List of gateways for Company A](image-url)
8. Select **Account Admin → Profiles → B2B Capabilities** so that Company E’s WebSphere BI Connect server knows that Company A can handle AS2 EDI documents. The procedure is similar to the one for Company E. The result is shown in Figure 7-34.

![Figure 7-34](image)

*Figure 7-34  B2B capabilities for Company A as stored on server of Company E*
Before you proceed to the last step, again refer to the overview pictures for the outbound flow. In 7.3, “Configuration tasks by Company E administrator” on page 148, we defined the B2B capabilities for Company E, which is referenced in Figure 7-35 and in Figure 7-36. We also defined the FileSystemGateway object, which is also marked in Figure 7-35.

Figure 7-35  Status of the configuration for inbound flow
In 7.4, “Configuration tasks for partner Company A” on page 156, we defined the B2B capabilities for Company A, which is referenced in Figure 7-35 and in Figure 7-36. We also defined the HTTPGateway object, which is marked in Figure 7-36.

Combining the previous summary with the current, there is only one concept to complete in each picture: the participant connection that ties together source and target participant, B2B capabilities, and the different gateways.

![Diagram](image-url)

*Figure 7-36  Status of the configuration for outbound flow*
7.5 Connecting Company E to Company A

At this point, the administrators of Company E and A have performed their tasks on the WebSphere BI Connect server. Now the hub administrator can tie the pieces together and create the participant connection.

Actually, this task can be performed by the administrator user of the community manager of Company E. This person can also tie everything together in a single connection. From a role-based view, it may be better to assign this task to the hubadmin. As soon as the participant connection is created, activity may happen in the server. The hubadmin has to look after the server. Also, it gives the hubadmin the opportunity to review the configuration that was created by the admin users of both Company A and Company E.

The following discussion describes the required steps while logged on as hubadmin (Figure 7-37).

Figure 7-37  Logging on as hubadmin to the server of Company E
1. Log on as hub administrator.
2. Select **Account Admin → Participant Connections**. See Figure 7-38.
3. The browser presents the option to manage connections. For Source, select **Company E**, and for Target, select **Company A**. Click **Search**.

*Figure 7-38  Searching for a connection between Company E and Company A*
4. WebSphere BI Connect now looks for matching B2B capabilities between Company A and Company E. Figure 7-39 shows the matches. Click **Activate** to activate the matching connections.

*Figure 7-39  Matching B2B capabilities*
5. When the connections are activated, more options are shown (Figure 7-40). Click **Gateways** to select the correct gateway for this connection.

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-40  Connection details**
A window opens where the correct gateways are already selected since we defined the gateways as the default gateways (Figure 7-41). Review the gateway information and click **Save**.

![Figure 7-41](image)

*Figure 7-41  Confirming the selected gateways*

Click **Attributes** next to the AS2 package for the target partner (Figure 7-40). This allows you to set a number of AS2-related parameters.
6. Figure 7-42 shows the current connection attributes. As you can see, there is currently no value for the parameter AS MDN Http Url. However, under that parameter, you see another parameter, called AS MDN Requested, which is set to Yes. Clearly, this cannot work. Requesting a message disposition notification (MDN) means that you should also tell where that MDN needs to be delivered.

To update the parameter AS MDN Http Url, click the icon next to **Package: AS (N/A)**. This opens a folder where AS2-related parameters can be changed.

![Connection Attributes](image)

**Figure 7-42  Current connection attributes**
7. Figure 7-43 shows all the AS2 related parameters that can be updated. Provide a URL for the parameter AS MDN Http Url, for example:

http://wbichub:57080/bcgreceiver/companye/edi_in

Provide a value for the parameter AS MDN Email Address, and click **Save** (at bottom of the window shown in Figure 7-43).
8. Click **Return** at the top to return to the window to manage connections.

9. Back in the window to manage connections, select the reverse combination. That is, select **Company A** for Source and **Company E** for Target.

10. Activate the combination and review the gateway settings and connection attributes.

This completes the steps to set up WebSphere BI Connect Enterprise at Company E. You must perform a similar setup for WebSphere BI Connect Advanced at Company A, as discussed in the following section.

### 7.6 Configuration tasks for hubadmin of Company A

The setup of WebSphere BI Connect Advanced on AIX is not different from the setup of WebSphere BI Connect Enterprise on Windows. In this section, we no longer provide detailed instructions on how to configure WebSphere BI Connect to complete the scenario of this chapter. The interface and the menu options are exactly the same (Figure 7-44).

**Note:** Figure 7-43 shows a number of parameters to control encryption and digital signatures. These features are currently not activated in our setup. Do not activate them at this time, since we have not discussed the use of certificates and encryption keys. Chapter 8, “Securing the B2B exchange” on page 189, explains how to use encryption and digital signatures.
As for Company E, the hub administrator of Company A needs to log on to the WebSphere BI Connect server to:

- Define targets
- Create interactions
- Define community manager and community participant

Two targets need to be created, an HTTP target and a file system target. When we describe the task of the administrator of Company A on the server of Company E, we use `http://9.42.171.84:57080/bcgreceiver/companya/edi_in` to define the gateway at Company E. This is the value of the HTTP target that the hub administrator of Company A needs to create. Refer to Figure 7-32 on page 158.

The second target is the directory that WebSphere BI Connect monitors to send EDI documents. When creating this target, make sure that the user bcguser has full access to this directory. You must also ensure that all higher level directories exist and are accessible to the user bcguser.

For example, if you choose the `/home/bcguser/WBIConnect/data/companya/edi_out` directory for sending EDI documents, then you make sure that the `/home/bcguser/WBIConnect/data/companya` directory exists and is
accessible to bcguser. To avoid any security problems, create such a directory structure while logged on as bcguser to the AIX system.

After you create the target FileTarget /home/bcguser/WBICConnect/data/companya/edi_out, a list of targets is displayed as shown in Figure 7-45.

![Figure 7-45   List of targets for Company A](image)

The creation of interactions is exactly the same as explained in 7.2.2, “Creating interactions” on page 132. We assume for a moment that the AIX server has the same requirements as the Windows server.

Then, the hub administrator at Company A can proceed to define the community manager for this implementation of WebSphere BI Connect.

The company name is set to companyA and the business identifier is set to companya. The value companya as a business identifier was also used in the setup of WebSphere BI Connect at Company E, so the values need to match.

After creating the community manager, the hub administrator can proceed by defining a community participant for Company E. Here again, you must set the business identifier, which was companye.
Thus, on the server owned by Company E, we have a community manager, representing Company E, and a community participant, representing Company A. On the server owned by Company A, we again have a community manager and a community participant. However this time, the roles are reversed.

Figure 7-46 shows the list of participants that are defined to WebSphere BI Connect owned by Company A.

![List of participants at WebSphere BI Connect for Company A](image)

At this time, the newly defined administrator for Company A can log on to their own WebSphere BI Connect server and add their B2B capabilities to the system.
The administrator also needs to create a gateway where WebSphere BI Connect can deliver incoming EDI documents (Figure 7-47). Remember that this directory needs to exist, including the lowest level directory (edi_in in our setup). As before, the directory needs to be accessible to bcguser. The administrator of Company A creates the file system gateway with the value file:///home/bcguser/WBIConnect/data/companya/edi_in.

![Figure 7-47 Creating the file system gateway](image)

Now, the administrator for Company E can log on to the WebSphere BI Connect server owned by Company A and log on as a regular community participant. After resetting their generated password, he then can provide his B2B capabilities and create the gateway that points to his own WebSphere BI Connect server.
The value of this gateway, listed in Figure 7-48, needs to be the same as the value for the HTTP target at their own server (see Figure 7-6 on page 129).

![Figure 7-48 Creating an HTTP gateway](image)

When the administrators of Company A and Company E perform their tasks on the server owned by Company A, the hub administrator can log on again and perform the last step: to make participant connections between both partners.
Figure 7-49 shows the activated connections where Company A is the source. You also need to activate the reverse connections, as explained in 7.5, “Connecting Company E to Company A” on page 163, which describes the configuration of WebSphere BI Connect for Company E.

After activating the connections, review the gateways again and update the following connection parameters:

- AS MDN Http Url
- AS MDN Email Address
Figure 7-50 shows the update process for these attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Current Value</th>
<th>Inheritance</th>
<th>Update</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time To Acknowledge</td>
<td>Time To Acknowledge</td>
<td>30</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: Time To Acknowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retry Count</td>
<td>Retry Count</td>
<td>3</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: Retry Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Compress Before Sign</td>
<td>AS Compress Before</td>
<td>Yes</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sign</td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS Compress Before</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Compressed</td>
<td>AS Compressed</td>
<td>No</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS Compressed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Encrypted</td>
<td>AS Encrypted</td>
<td>No</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS Encrypted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Http Url</td>
<td>AS MDN Http Url</td>
<td>No value provided</td>
<td>No value provided</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Email Address</td>
<td>AS MDN Email Address</td>
<td>No value provided</td>
<td>No value provided</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Asynchronous</td>
<td>AS MDN Asynchronous</td>
<td>Yes</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS MDN Asynchronous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Requested</td>
<td>AS MDN Requested</td>
<td>Yes</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS MDN Requested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Message Digest</td>
<td>AS Message Digest</td>
<td>sha1</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Algorithm</td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS Message Digest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Signed</td>
<td>AS MDN Signed</td>
<td>No</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: Global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS MDN Signed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Signed</td>
<td>AS Signed</td>
<td>No</td>
<td>Inherited from:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Scope: global</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type: AS Signed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Business Id</td>
<td>AS Business Id</td>
<td>No value provided</td>
<td>No value provided</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7-50  Connection attributes

This completes the setup for the WebSphere BI Connect server owned by Company A. Now you can test the connection and try to exchange documents between both partners.
7.7 Validating communication

Before trying a sample exchange between the two companies, the hubadmin user has one more tool that they can use to perform some validation. This tool can verify if WebSphere BI Connect is able to reach the partner’s WebSphere BI Connect server.

1. To access this tool, select **Tools → Test Participant Connection**.
2. A number of selection options are shown in this tool (see Figure 7-51). Select the target participant and the gateway.
3. Select the **POST HTTP** command.
4. WebSphere BI Connect should load the URL. Click **Test URL** to verify the connection.

If all goes well, an HTTP status code of 200 should be returned.

![Test Participant Connection](image)

*Figure 7-51  Validating communication from Company E to Company A*

End-to-end validation is only possible by dropping an EDI file in the correct directory, verifying that it arrives at your partner’s machine, and verifying that the MDN was received as well.
Example 7-1 shows a sample 850 EDI document, which is a purchase order. The Interchange Sender ID and Interchange Receiver ID in the ISA segment match the Business IDs as defined in the participant profiles.

Example 7-1    Sample EDI file to be sent to Company A

ISA*00*ssssssss*00*rrrrrrrrr*companye* *companya* *961007*2013*U*00200*000000001*0*T*:  
GS*P0*S1S1S1S1S1S1S1S1S1S1RIRIRIRIRIRIRIRIR*961007*2013*000000004*X*003050  
ST*850*000040001  
BEG*00*BE*2a*43324234v5523*961007*23tc4vy24v2h3vh3vh*ZZ*IEL*09*RE*09  
CUR*11*TRN*5656*65*656*IMF*006*961007  
REF*6A*433r1c3r34r34c3312qctgc54*Reference Number  
PER*AA*Hans Gutten*CP*1.322.323.4444*****rgg4egv4t4  
TAX*4tgtbt4tr4tr*GL*ghgh*********G+C  
FOB*TP*CA*USA*02*DOM*CC*Regular Locations per Terms  
CTP*DE*C04*25000*D9*SEL*23214*23432423423*ES*42243423  
SAC*A*B000*AE*3545*3442300  
CUR*11*767*7767*65  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
P01*111-aaa*1000000*AS*90.00*BD*AK*234235v3534q6f3534v4353453vq3q32************A1*ytrh  
CTT*1  
SE*22*000040001  
GE*1*000000004  
IEA*1*000000001

When an ANSI X12 EDI file is placed in the directory \WBIConnect\data\companye\edi_out\Documents\Production on the server of Company E, the following events take place:

1. The WebSphere Bi Connect Enterprise Receiver polls this directory for new messages.
2. When the message is received, it is picked up by the Receiver and placed elsewhere in the file system for processing by the WebSphere Bi Connect Document Manager.
3. The Document Manager parses the EDI envelope details to identify the participants involved in the transaction.
4. When the sender and receiver identifiers are determined, the Document Manager looks up the participant connection and builds the AS2 header.
5. The message is then sent to the gateway employed by the participant connection.

6. In this scenario, our gateway is defined as using HTTP. This gateway points to the HTTP target of WebSphere BI Connect of Company A.

7. The WebSphere BI Connect Receiver of Company A listens for inbound messages sent over HTTP.

8. Again, the Receiver picks up the message and places it on the file system for processing by the Document Manager.

9. The Document Manager parses the MIME headers of the AS2 packaging and the EDI envelope data to determine the participants involved.

10. When the participants are known, the Document Manager determines the participant connection to use.

11. In our case, the participant connection instructs the Document Manager to strip the AS2 packaging and place the document in the file system.

12. The EDI transaction is now seen in the /home/bcguser/WBIConnect/data/companya/edi_in directory. The document file name is not preserved in the transmission. A typical file name upon receipt is 108994088762092AAB5451bcab061a0716ba029abfa370fcd56c99d9097fc4.vcm.

In addition to simply looking at the edi_in directory on the partner’s system, you can use several tools within WebSphere BI Connect to review the communication and the exchange of documents. Chapter 12, “Managing the B2B exchange” on page 337, discusses the options that are available in the product to review the communication.

For now, we focus on the AS1/AS2 Viewer. This tool is available for the hubadmin user, the admin user of the community manager, and the admin user of the community participant. However, as community participant, you can use the AS1/AS2 viewer only for communications that involve your own company. But, it is possible to log on to your partner’s server and to use their AS1/AS2 viewer to determine what has happened on their server with documents that you sent to them, for example.

To use the viewer, select Viewers → AS1/AS2 Viewer in the menu bar. First, you can provide filters to limit the output in the viewer. For example, you can specify a time range, limit the search to a combination of partners, or limit it to a certain gateway type (such as production or test). The default search options result in a list of all AS1/AS2 exchanges during the last hour.

Click Search, and you see a list of AS1/AS2 exchanges that match the search arguments. Click the icon next to the AS1/AS2 communication that interests
you. This results in the details of that exchange. An example is shown in Figure 7-52.

The top section lists the main details about this exchange:

- Message ID
- Source and Target Participant
- Time Stamp
- Gateway information
- MDN information

Below that, you see the incoming EDI document (package None) and packaged as AS. The lowest section covers the MDN, which is sent by Company A to Company E. Therefore, the roles of Company A and Company E are now reversed and Company A is listed as the source.

![WebSphere Business Integration Connect Community Console](image)

**Figure 7-52 Details of the AS2 exchange**
From within Figure 7-52, you can review the source document, the document as it is sent to the partner, and the MDN document. To review the HTTP and AS2 header information, click the icon for the AS packaged document. A window opens that shows the actual details. See Figure 7-53.

![Figure 7-53 AS packaged EDI document](image-url)
In the same way, you can inspect the MDN that was sent by Company A. See Figure 7-54 for the MDN matching with the AS2 document shown in Figure 7-53.

![Figure 7-54](image-url)

**Figure 7-54**  MDN received from Company A
7.8 Revisiting role-based configuration

During the configuration of WebSphere BI Connect on both machines in the previous sections, we stressed the role-based approach and what has to be done by each user. However, this is not the only approach that can be taken. The user hubadmin can perform all roles and edit the profile of both participants himself.

After you log on as hubadmin, you see the Participant Search window. Click Search to display an unfiltered list of defined participants (Figure 7-55).

![Figure 7-55 Searching for participants](image)
When the list of defined participants is shown, click the icon in Figure 7-56 next to the profile of Company A, for example. From that point on, the hubadmin works with the profile of that participant. It is as though the hubadmin acts on behalf on the user admin of Company A.

![Figure 7-56   List of participants](image-url)
When the profile is shown (Figure 7-57), the hubadmin can now view and edit all elements of the profile of Company A. For example, as the admin user of Company A, you had to define your gateways.

Figure 7-57  Profile of Company A inspected by the hubadmin user
If the user hubadmin selects the Gateways menu option in Figure 7-57, they see the gateway that was defined by the user admin of Company A. The hubadmin can create additional gateways or change the properties of a gateway by clicking the icon next to the gateway (see Figure 7-58).

However, the Gateways menu option in Figure 7-58 is completely different from the Gateways menu option in Figure 7-55 on page 184. The first option is for gateways that belong to the participant Company A, while the last option gives access to the gateways that belong to the virtual participant Operator. You should not define gateways in the profile of Operator, since this participant does not interact with any internal system or with any business partner. The virtual participant Operator should be used to manage the overall server and not interact with business partners.
Securing the B2B exchange

Chapter 7, “Creating a basic B2B exchange” on page 121, describes the implementation of the exchange of electronic data interchange (EDI) documents via AS2. However, this exchange lacked the usual security aspects of a typical AS2 configuration. This chapter shows you how to add encryption and digital signatures so that the exchange of documents is performed in a secure way.
8.1 What is needed to perform encryption and decryption

Encryption is a generic term to describe a collection of algorithms that scramble data so that only the intended receiver of the data can unscramble or decrypt the data. As a basic requirement, encryption is a reversible process. This is not a characteristic of digital signatures. The creation of a digital signature results in a stream of bytes from which you cannot rebuild the original text.

Encryption algorithms can roughly be categorized in two classes. The first class uses symmetric algorithms. Symmetric key algorithms require the key to be shared between two parties before encryption can take place. This creates an opportunity for the key to be intercepted and used by a third party to eavesdrop on the secure communications. A good example of a symmetric algorithm is DES or Triple DES.

The second class uses asymmetric algorithms or public key algorithms. Public key cryptography, by contrast, allows the public part of the key pair to be publicly available. This is because it is impossible to determine the private key based on the public key.

For example, if Alice wants to send an encrypted message to Bob, Alice looks up Bob’s public key, encrypts the message, and sends it to Bob. Bob then decrypts the message using his private key. Since no secret information needs to be shared between the two parties, public key cryptography is considered more secure. On the down side, public key cryptography is much more computation intensive than symmetric key cryptography. Typically a mixture of the two types is used in practice.

A public key system is used to exchange a one-off symmetric key that the parties use to encrypt their subsequent conversation. A temporary session key is generated by the sender. It is sent to the receiver after encryption with the public key of the receiver using public key cryptography. Thus, the session key can only be obtained by the receiver, who is the owner of the matching private key. The actual data is encrypted using the temporary key via a symmetric algorithm. This two-step process is shown schematically in Figure 8-1. The fact that the use of different keys still results in a reversible process is a consequence of the fact that these algorithms are built upon a prime number theory. The generation of public and private keys relies on the knowledge of large prime numbers.
The main advantage of public key cryptography is its level of security. The parties that exchange documents using these algorithms do not share a common piece of data, which reduces the risk of security breaches. The main disadvantage of public key cryptography is performance. In a B2B environment, it is not uncommon to send large documents of several megabytes. Encrypting these kinds of documents using public key cryptography requires a significant amount of CPU resources.

Figure 8-1 Encrypting EDI document using symmetric/asymmetric algorithms
When the business partner receives the encrypted document, first they extract the encrypted session key and decrypt it with their private key. When they obtain the session key, they use it to decrypt the actual EDI document, as shown schematically in Figure 8-2.

Since the session key can be obtained only by the owner of the private key, the EDI document can be decrypted only by the intended receiver.

*Figure 8-2  Decrypting the session key and EDI document*
Let us now return to the configuration of WebSphere BI Connect for Company A and Company E (Figure 8-3). When sending an EDI doc over AS2 from Company E to Company A, the following arrangement is required:

- Company E needs the public certificate of Company A to encrypt.
- Company A uses its own private certificate to decrypt.

**Figure 8-3 Scenario overview**
8.2 Enabling encryption

Enabling encryption for AS2 consists of several steps:

1. Generate a private/public key pair.
2. Upload the private key to the server that is owned by the private key owner, so that the server can decrypt documents.
3. Upload the public certificate to the server of the trading partner, so that the documents can be encrypted before being sent to the owner of the public certificate.
4. Update the AS2 connection parameters to request encryption.

These steps are required for both partners if you want to send and receive encrypted documents over AS2 between the two partners.

8.2.1 Company E generates a public/private key pair

WebSphere BI Connect provides a tool to generate a public/private key pair. This tool, called ikeyman, can be used to create self-signed certificates. However, it is quite common to request certificates from an external organization that is considered trustworthy by the business partners. Such an external organization is sometimes called a certificate authority or root certificate authority.

To open the ikeyman tool, navigate to the <install_dir>receiver\was\bin directory and execute the ikeyman.bat file.

When the ikeyman tool opens, generate a new self-signed certificate:

1. From the Key Database File menu, select Open.
2. In the Open window (Figure 8-4), complete these tasks:
   a. Click Browse.
   b. Navigate to the C:\WBConnect\common\security\keystore\ directory and select the receiver.jks file. Click Open.
   c. In the Open window, click OK.
3. At the Password prompt, enter WebAS.

4. In the IBM Key Management window (Figure 8-5), delete the WebSphere dummy server certificate.

5. Click **New Self-Signed**.
6. In the Create New Self-Signed Certificate window (Figure 8-6), complete the details of the new certificate:
   a. Provide a label for the key, for example, CompanyE_SelfSigned.
   b. Provide a common name, for example, wbichub.itso.ral.ibm.com.
   c. Provide a name for the organization, for example, Company E.
   d. Provide values for the optional fields if applicable.
   e. Click OK to generate the certificate.

![Create New Self-Signed Certificate](image)

Figure 8-6 Creating a new self-signed certificate

Next you must extract a public certificate from the new self-signed certificate. The WebSphere BI Connect Advanced server of Company A uses this public certificate to encrypt messages that are sent to the WebSphere BI Connect Enterprise of Company E. This means that at some point you need to upload this public certificate to the server of our partner. Since only the owner of the private key, Company E, can decrypt the messages sent by Company A, we know then that confidentiality is assured (Figure 8-7).
1. In the IBM Key Management window (Figure 8-7), click **Extract Certificate**.

![IBM Key Management window](Image)

*Figure 8-7  Extract certificate of key store*

2. In the Extract Certificate to a File window (Figure 8-8), complete these tasks:
   a. For Data type, select **Binary DER data**.
   b. For Certificate file name, type `CompanyEPublicCertificate.der`.
   c. Click **OK**.

![Extract Certificate to a File window](Image)

*Figure 8-8  Extract certificate of key store*

The public certificate is extracted in a form that can be used by WebSphere BI Connect.
Now you must export a private certificate from the newly created self-signed certificate. WebSphere BI Connect Enterprise of Company E uses this to decrypt encrypted transactions.

1. From the ikeyman tool (Figure 8-5 on page 195), click **Export/Import**.
2. In the Export/Import Key window (Figure 8-9), complete these tasks:
   a. For Key file type, select **PKCS12**.
   c. Click **OK**.

*Figure 8-9  Export a key*
3. Since this file contains your private key, you must protect it very well. To avoid unauthorized use, the contents of this file are protected by a password. Any person who wants to use this file in an encryption/decryption solution has to know this password before such an encryption/decryption solution accepts the file.

In the Password Prompt window (Figure 8-10), enter a password to protect the target PKCS12 file and click **OK**. Be sure to make a note of this password, because it is required later.

![Password Prompt](image)

*Figure 8-10  Provide a password to protect the private key*

You are now finished working with the key management utility and can use the exported certificates with WebSphere BI Connect.

### 8.2.2 Company E uploads private key to its own server

You must now import the private key file into the WebSphere BI Connect Enterprise server that is owned by Company E. Remember that this private key is used when decrypting documents sent by the partners of Company E, such as Company A.
To upload the PKCS12 file, follow these steps:

1. Log in to WebSphere BI Connect as the Hub Admin.

2. Select **Account Admin → Profiles → Certificates**.

3. Select **Load PKCS12** as shown in Figure 8-11.

4. In the Create New PKCS12 Certificate panel (Figure 8-12), complete these tasks:

   a. For Certificate Type, select **Encryption**.

   b. Provide a description for the certificate.

   c. For Status, select **Enabled**. Do not forget this, since by default the status is set to disabled.

   d. Click **Browse** and navigate to the directory in which the PKCS12 file is stored, for example `C:\WBIConnect\common\security\keystore\`. Select the file and click **Open**.

   e. Enter the password used when exporting the PKCS12 file earlier (see Figure 8-10).

   f. Click **Upload**.
Figure 8-12 Loading the private key file
5. The server now reviews the certificate and presents information about it, as shown in Figure 8-13. Review this to make sure that the information is correct. Then click Save.

![Profile: Hub Operator: Create New PKCS12 Certificate](image)

**Certificate Information**

**General Information**

- **Status**: This certificate is valid
- **Issued To**: wbcibhub.itso.rational.ibm.com
- **Valid**: 7/19/04 through 7/19/06

**Detail Information**

- **Version**: 3
- **Serial Number**: 007C A009
- **Valid From**: Monday, July 19, 2004 6:18:33 PM GMT
- **Valid To**: Tuesday, July 19, 2005 6:18:33 PM GMT
- **Fingerprint**: MD5
- **Algorithm**: SHA-1
- **Public Key**: 6776 F43F 7249 E90F F23A 7C33 7218 7F2E 5F37 8F4B 9232 4F4D 5D60 89E2 24F4 537A 7192 88E2 B17D 0E90 0100 01

**Subject Information**

- **Common Name (CN)**: wbcibhub.itso.rational.ibm.com
- **Organization**: Company E
- **Locality (L)**: US
- **State (ST)**: US

*Figure 8-13  Saving the uploaded certificate after review*
6. Click **List** to return to the list of certificates that now contains the new entry. You see the list in Figure 8-14.

![Certificate uploaded](image)

**Figure 8-14 Certificate uploaded**

At first, it may seem odd that the hub administrator has to upload the private key of Company E. Intuitively, you would expect that the administrator of Company E performs that task. However, if you think about the actual processing that happens when an encrypted file is received, it sounds logical that the hub administrator owns the private certificate.

When an encrypted document is received on an HTTP target, the WebSphere BI Connect server cannot know the target participant because the document is encrypted. The server needs to look for the certificate in a generic place, such as the operator profile. When the document is decrypted, the server determines what the target business entity is.
8.2.3 Company E uploads public certificate to partner’s server

The next step is to upload the public certificate to the partner’s server. This time, the administrator of Company E logs on to the server of Company A. Company A uses this key to encrypt any documents targeted for Company E.

1. Log in as the Community Participant on the partner’s server of Company A.
2. Select **Account Admin** → **Profiles** → **Certificates**.
3. In the Certificate List panel (Figure 8-15), click **Load Certificate**.

![Figure 8-15  List of your certificates on your partner’s system](image)
4. In the Create New Certificate panel (Figure 8-16), complete these tasks:
   a. For Certificate Type, select **Encryption**.
   b. Provide a description for the certificate.
   c. For Status, select **Enabled**.
   d. Click **Browse** and navigate to the directory where you have stored the public certificate, for example C:\WBICConnect\common\security\keystore\. Note that this is a folder on the machine where you use the browser, not on the partner’s machine.
   e. Click **Upload**.

![Figure 8-16  Uploading the certificate to the server of your partner](image-url)
5. The certificate is now analyzed by the server of your partner. Review this analysis as shown in Figure 8-17, and click Save.

![Certificate Information](image)

**Figure 8-17  Saving the uploaded certificate**
8.2.4 What happens next?

At this moment, Company A can send an encrypted document to his partner Company E. At this stage, you can amend the participant connection between Company A and Company E on the server of Company A and indicate that encryption is required. However, given that we want encryption in both directions for EDI documents, we enable encryption from the perspective of Company A.

8.2.5 Company A generates a public/private key pair

The process that needs to be executed from a Company A perspective is similar to the process for Company E. For completeness, this section explains these steps.

First, the files that we need to use (create, update, or both) are either owned by the user ID bcguser or should be owned by bcguser. Therefore, it is important that you run the key management tool while logged on with this user.

1. Launch the key management tool via the script ikeyman.sh, which can be found in /opt/IBM/WBIConnect/receiver/was/bin.
2. When the key management tool is launched, select Key Database File → Open.
3. In the Open window (Figure 8-18), click Browse and navigate to the directory /opt/IBM/WBIConnect/common/security/keystore and open the receiver.jks file. Click OK.

![Figure 8-18 Opening the existing key store](image)

4. At the password prompt, enter WebAS.
5. Delete the WebSphere dummy server certificate.
6. In the IBM Key Management window (Figure 8-5 on page 195), click New Self-Signed.
7. In the Create New Self-Signed Certificate window (Figure 8-19), complete the details of the new certificate:

   a. Provide a label for the key, for example, CompanyA_SelfSigned.
   b. Provide a common name, for example, m106984f.itso.ral.ibm.com.
   c. Provide a name for the organization, for example, Company A.
   d. Click OK.

   ![Create New Self-Signed Certificate](image)

   **Figure 8-19 Creating a new self-signed certificate**

   The next step is to extract a public certificate from the new self-signed certificate. The WebSphere BI Connect Enterprise server of Company E uses this public certificate to encrypt messages that are sent to the WebSphere BI Connect Advanced of Company A. This means that at some point, you need to upload this public certificate to the server of the partner. Since only the owner of the private key, that is Company A, can decrypt the messages sent by Company E, we know then that confidentiality is assured (Figure 8-20).
1. In the IBM Key Management window (Figure 8-20), click **Extract Certificate**.

![IBM Key Management window](image)

*Figure 8-20   Selecting Extract Certificate*

2. In the Extract Certificate to a File window (Figure 8-21), complete these tasks:
   
   a. For Data type, select **Binary DER data**.
   
   b. For Certificate file name, type `CompanyAPublicCertificate.der`.
   
   c. Click **OK**.

![Extract Certificate to a File window](image)

*Figure 8-21   Extracting the certificate of key store*

This public certificate is extracted in a form that can be used by WebSphere BI Connect.
Next you generate a private certificate from the newly created self-signed certificate. This is used by WebSphere BI Connect Advanced of Company A to decrypt encrypted transactions.

1. From the ikeyman tool, click **Export/Import**.

2. In the Export/Import Key window (Figure 8-22), complete these tasks:
   a. For Key file type, select **PKCS12**.
   b. For File name, type `CompanyAPrivateCertificate.p12`.
   c. Click **OK**.

   ![Figure 8-22 Export a key](image)

3. Since this file contains your private key, you must protect it well. To avoid unauthorized use, the contents of this file are protected by a password. Any person who wants to use this file in an encryption/decryption solution must know this password before such an encryption/decryption solution accepts the file.

   In the Password Prompt window (Figure 8-23), enter a password to protect the target PKCS12 file and click **OK**. Be sure to make a note of this password, because it is required later.

   ![Figure 8-23 Providing a password to protect the private key](image)
8.2.6 Company A uploads a private key to its own server

Now you import the private key file into the WebSphere BI Connect Advanced server that is owned by Company A. Remember that this private key is used when decrypting documents sent by the partners of Company A, such as Company E.

When using a browser to upload a file stored on AIX to the WebSphere BI Connect server running on AIX, the browser should be running on AIX as well. The browser opens the file and sends it to the Web server and to WebSphere BI Connect. It is not WebSphere BI Connect that reads the file based on the contents of the entry field in the browser.

Given that, you either need to use a browser running on AIX or download the certificates from AIX to a PC platform and run the browser on the PC. The following instructions assume that the files were downloaded to a PC and stored in C:\WBICTempAIX.

To upload the PKCS12 file, follow these steps:
1. Log in to WebSphere BI Connect as the Hub Admin.
2. Select Account Admin → Profiles → Certificates.
3. Select Load PKCS12.
4. In the Create New PKCS12 Certificate window (Figure 8-24), do these tasks:
   a. For Certificate Type, select **Encryption**.
   b. Provide a description for the certificate.
   c. For Status, select **Enabled**.
   d. Click **Browse** and navigate to the directory in which the PKCS12 file is stored, for example C:\WBICTempAIX. Select the file and click **Open**.
   e. Enter the password used when exporting the PKCS12 file earlier (see Figure 8-23).
   f. Click **Upload**.

*Figure 8-24  Uploading the private key to your own server*
5. The server now reviews the certificate and presents you with information about it as shown in Figure 8-25. Review this to make sure that you upload to the server what should be uploaded. Then click **Save**.

![Figure 8-25  Saving the uploaded certificate](image-url)
6. Click **List** to return to the list of certificates that now contains the new entry. See Figure 8-26.

![Certificate uploaded](image)

**Figure 8-26 Certificate uploaded**

### 8.2.7 Company A uploads public key to the partner’s server

Upload the public key to the partner’s server. This time, the administrator of Company A logs on to the server of Company E. Company E uses this key to encrypt any documents targeted for Company A.

1. Log in as the Community Participant on the partner’s server of Company E.
2. Select **Account Admin** → **Profiles** → **Certificates**.
3. Click **Load Certificate**.
4. In the Create New Certificate window (Figure 8-27), complete these tasks:
   a. For Certificate Type, select **Encryption**.
   b. Provide a description for the certificate.
   c. For Status, select **Enabled**.
   d. Click **Browse** and navigate to the directory where you have stored the public certificate, for example C:\WBICTempAIX. This is a folder on the machine where you use the browser, not on the partner’s machine.
   e. Click **Upload**.

![Image of the Create New Certificate window with options for Certificate Type, Description, Status, and Browse for Certificate location]

*Figure 8-27  Uploading the certificate to the server of your partner*
5. The certificate is now analyzed by the server of your partner. Review this analysis, which is shown in Figure 8-28, and click **Save**.

![Profile: Company A > Create New Certificate](image)

**Figure 8-28 Saving the uploaded public certificate**

### 8.2.8 Updating the participant connections

To configure the use of encryption for an AS2 exchange between two partners, you need to change the settings of the participant connection. Earlier 7.5, “Connecting Company E to Company A” on page 163, explained how to inspect and change connection attributes for AS2 when sending documents from Company E to Company A.
Figure 8-29 shows the change that is required. To enable encryption, select **Yes** next to the label AS Encrypted and click **Save**. From now on, WebSphere B2B Connect generates a random session key, encrypts the session key using the public key of the receiver, and uses the session key in a symmetric encryption algorithm to encrypt the actual data.

You must make the same changes to the participant connection from Company A to Company E. Refer to 7.6, “Configuration tasks for hubadmin of Company A” on page 170, which explains the setup on the server of Company A and how to access the AS2 connection attributes. Change the setting AS Encrypted to **Yes** for that connection as well.
8.2.9 Validating that encryption is enabled

You can validate the encryption in the same way as before. You drop an EDI file into the directory that is polled by the WebSphere BI Connect Enterprise FileSystemTarget. It arrives in the corresponding FileSystemGateway on the WebSphere BI Connect server of Company A. The fact that the file arrives is an indication that the changes to the participant connection have not broken the setup from the end of Chapter 7, "Creating a basic B2B exchange" on page 121.

To verify that encryption has occurred, again you use the AS1/AS2 Viewer.

1. From the menu bar, select **Viewers → AS1/AS2 Viewer**.
2. Provide search arguments and click **Search**.
3. When you see the list of AS1/AS2 sessions matching the search criteria, click the icon to view the details of the selected AS2 exchange (Figure 8-30).

Assuming that the same EDI file is used, you see that the file size quoted in Figure 8-30 is higher than the file size quoted in Figure 7-52 on page 181. This is already an indication that something different has happened.

![Figure 8-30 Details of an AS2 communication with encryption](image)
To inspect the raw document, click the icon for the AS packaged document sent by Company E. This raw document is shown in Figure 8-31.

Clearly, the EDI document is not readable anymore. The content type in Figure 8-31 is set to application/pkcs7-mime. Compare this with Figure 7-53 on page 182, where the EDI document is shown in clear text and the content type is set to application/edi-x12. This proves that the document was encrypted.

![Figure 8-31 Raw document viewer for encrypted document](image)

### 8.3 What is needed to digitally sign and verify the signature?

Encryption results in a confidential exchange of information. Only the owner of the private key can read the message.
However, as the recipient of that document, there are still two open issues:

- Is this document really sent by our business partner? Anyone that has obtained the public key of the receiving company could have created this document.

- Is the received document exactly the same as the sent document? It could have been intercepted and altered somewhere between sender and receiver.

Adding a digital signature to the document provides non-repudiation and message integrity. In general, digital signatures can be used independently of the encryption. But in many B2B environments, both technologies are combined.

First a message digest of the document is created using a one-way hashing algorithm such as SHA1. Then the message digest is encrypted using the sender’s private key. This is a signature. The signature is sent with the document, as shown in Figure 8-32.

![Figure 8-32 Signing a document by calculating a message digest](image)

When the business partner receives the document, they decrypt the message digest with the sender’s public key and compare the decrypted message digest with one that is calculated from the actual document. If there is a match, the
receiver knows that this is the document the sender intended to send and that
the sender was the one that sent it. This process is shown schematically in
Figure 8-33.

The authenticity of the received document relies on the authenticity of the public
key, meaning is the public key really owned by the organization that pretends that
owns it. That guarantee is provided by the signer of the public key, which is
usually a root certificate authority. To be complete, we should add another step to
Figure 8-33. The public key of the sender is validated by using the public key of
the signing authority. When using the services of a certificate authority, you need
the public key of that authority and to make that public key available to
WebSphere BI Connect. For self-signed certificates, the public key of the sender
is used in both roles:

- Validate itself
- Calculate the message digest

The fact that the public key for self-signed certificates is used twice has its impact
on the configuration of WebSphere BI Connect.

When sending an EDI document over AS2 from Company E to Company A:

- Company E needs the public certificate of Company A to encrypt.
- Company E needs its own private key to sign it.
- Company A uses the public certificate of Company E to validate the signature
  of Company E.
- Company A uses its own private certificate to decrypt.
When sending an MDN from Company A to Company E:

- Company A needs its own private key to sign the MDN.
- Company E uses Company A's public certificate to validate the signature.

### 8.4 Enabling digital signatures

To enable digital signatures, you need again a public/private key pair. You can use the same key pair that was used for encryption. It is then sufficient to indicate in WebSphere BI Connect that the keys have an additional role.

#### 8.4.1 Changes to be performed on the server of Company A

You must perform the following four tasks:

1. Update the role of the private key of Company A.
2. Update the role of the public key of Company E that is stored in the profile of Company E.
3. Upload the public key of the signer of the public key of Company E to the profile of hubadmin. Since the public key of Company E is self-signed, you need to upload the public key of Company E in the profile of hubadmin. Thus, the public key for Company E is stored in their own profile on Company A and in the system profile of hubadmin.
4. Update the participant connection to activate signatures for AS2.

Follow the steps as explained here:

1. While logged on as hubadmin of Company A, select **Account Admin → Profiles → Certificates**.
2. The console should show the private key of Company A that was uploaded before and that has one single role: encryption. Click the view icon to open the details.
3. When the certificate is shown, click the edit icon and select the role **Digital Signature**. Click **Save**.
4. Return now to the list of certificates, which should now list both roles for the private key of Company A, as shown in Figure 8-34.

![Figure 8-34 List of certificates for hubadmin on Company A](image)

5. While logged on as hubadmin, select **Account Admin → Profiles → Community Participant**.

6. Click **Search** to display the list of profiles. Select the profile of Company E.

7. When the details of Company E’s profile are shown, click **Certificates** in the menu bar. Open the details of the public key of Company E and change the attribute Certificate Type to include digital signatures.

   **Note:** Updating the profile of Company E on the server of Company A can also be performed by the admin user of Company E logging on the server of Company A.

8. Upload the public certificate of Company E as root authority. This task has to be performed by the hubadmin of Company A. Select **Account Admin → Profiles → Certificates**.

9. When the list of certificates is shown, click **Load Certificate**.
10. In the Create New Certificate window (Figure 8-35), complete these steps:
   a. For Certificate Type, select **Root Certificate**.
   b. Provide a description.
   c. Set Status to **Enabled**.
   d. Click **Browse** to navigate to the actual file that contains the public key.
   e. Click **Upload**.

*Figure 8-35  Uploading the public key of Company E to Company A*
11. Review the analysis of the public key by WebSphere BI Connect, click Save. You should see the list of certificates for hubadmin on Company A as shown in Figure 8-36.

![Image of WebSphere BI Connect console](image)

**Figure 8-36  List of certificates for hubadmin on Company A**

Note: This last task may be considered as a task that is not in line with the role-based approach that we have used in this redbook so far. The hubadmin needs to have access to the file that contains the public key of their partner to upload it as a Root Certificate. Previously, when enabling encryption, the admin user of Company E could upload the file to the server of Company A and, as such, avoid a hand-over of their own public key file. This process is a direct result of the use of self-signed certificates.

In real-life situations, business partners usually require the use of certificates signed by a commercial certificate authority. In that situation, the hubadmin of Company A obtains the public key of the certificate authority directly from the certificate authority. Thus the hubadmin user can upload that public key to the server. In real-life situations, Figure 8-36 lists the private key of Company A and the public key of the certificate authority.
Table 8-1 summarizes what is uploaded in which profile and its use.

<table>
<thead>
<tr>
<th></th>
<th>Encryption</th>
<th>Decryption</th>
<th>Digital signature</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>hubadmin profile</td>
<td></td>
<td>Private key of Company A</td>
<td>Private key of Company A</td>
<td>Public key of Company E</td>
</tr>
<tr>
<td>Company A profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company E profile</td>
<td>Public key of Company E</td>
<td></td>
<td>Public key of Company E</td>
<td></td>
</tr>
</tbody>
</table>

12. While logged as hubadmin, select **Account Admin → Participant Connections**.

13. Select the source and target participants and click **Search**. Locate the connection for sending AS2 documents and click **Attributes**.

14. In the window shown in Figure 8-37, for AS Signed and AS MDN Signed, select **Yes**. Click **Save** to update the attributes.

It is possible to use only signatures on the actual AS document and not on the MDN. However, in practice, business partners require signatures on both the document and the MDN.
Figure 8-37  Updating AS2 connection attributes
8.4.2 Changes to perform on the server of Company E

Again you must perform the same four tasks:

1. Update the role of the private key of Company E.

2. Update the role of the public key of Company A that is stored in the profile of Company A.

3. Upload the public key of the signer of the public key of Company A to the profile of hubadmin. Since the public key of Company A is self-signed, you need to upload the public key of Company A in the profile of hubadmin. The public key for Company A is stored in his own profile on Company E and in the system profile of hubadmin.

4. Update the participant connection to activate signatures for AS2.

Since the tasks are exactly the same, we only show the results. Figure 8-38 shows the lists of certificates for hubadmin after performing the first and last task.

Figure 8-38 Uploading the public key of Company A to Company E

Figure 8-39 shows the list of certificates of Company A on the server of Company E after updating the role of the public key, which is the second task.
Again, the hubadmin of Company E or the admin user of Company A can perform this task.

Table 8-2 summarize where certificates are stored and what role they have.

<table>
<thead>
<tr>
<th>Description</th>
<th>Encryption</th>
<th>Decryption</th>
<th>Digital signature</th>
<th>Root</th>
</tr>
</thead>
<tbody>
<tr>
<td>hubadmin profile</td>
<td></td>
<td>Private key of Company E</td>
<td>Private key of Company E</td>
<td>Public key of Company A</td>
</tr>
<tr>
<td>Company A profile</td>
<td>Public key of Company A</td>
<td></td>
<td>Public key of Company A</td>
<td></td>
</tr>
<tr>
<td>Company E profile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 8-40 shows the changes that you need to make to the AS2 connection attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Current Value</th>
<th>Inheritance</th>
<th>Update</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time To Acknowledge</td>
<td>Time To Acknowledge</td>
<td>00</td>
<td>Inherited From: Scopes: Global Type: Time To Acknowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retry Count</td>
<td>Retry Count</td>
<td>3</td>
<td>Inherited From: Scopes: Global Type: Retry Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Compress Before Sign</td>
<td>AS Compress Before Sign</td>
<td>Yes</td>
<td>Inherited From: Scopes: Global Type: AS Compress Before Sign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Compressed</td>
<td>AS Compressed</td>
<td>No</td>
<td>Inherited From: Scopes: Global Type: AS Compressed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Encrypted</td>
<td>AS Encrypted</td>
<td>Yes</td>
<td>Locally Assigned</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>AS MDN Email Address</td>
<td>AS MDN Email Address</td>
<td><a href="mailto:mfr@companyx.com">mfr@companyx.com</a></td>
<td>Locally Assigned</td>
<td><a href="mailto:mfr@companyx.com">mfr@companyx.com</a></td>
<td></td>
</tr>
<tr>
<td>AS MDN Asynchronous</td>
<td>AS MDN Asynchronous</td>
<td>Yes</td>
<td>Inherited From: Scopes: Global Type: AS MDN Asynchronous</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Requested</td>
<td>AS MDN Requested</td>
<td>Yes</td>
<td>Inherited From: Scopes: Global Type: AS MDN Requested</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS Message Digest Algorithm</td>
<td>AS Message Digest Algorithm</td>
<td>ssh</td>
<td>Inherited From: Scopes: Global Type: AS Message Digest Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS MDN Signed</td>
<td>AS MDN Signed</td>
<td>No</td>
<td>Locally Assigned</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>AS Signed</td>
<td>AS Signed</td>
<td>No</td>
<td>Locally Assigned</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>AS Business Id</td>
<td>AS Business Id</td>
<td>No value provided</td>
<td>No value provided</td>
<td>Select one to update</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8-40   Updating the AS2 connection attributes
8.4.3 Validating that digital signatures are enabled

After you make the changes on both servers, you perform the validation in the same way.

1. Drop an EDI file in the target directory and verify that it still arrives in the gateway directory on the partner machine.

2. To verify that signatures have been used, you can again use the AS1/AS2 Viewer.

3. Locate again the AS2 transaction and open the details. As shown in Figure 8-41, the file sizes of the AS2 document and the MDN are again bigger than when using encryption only.

![webSphere Business Integration Connect Community Console](image)

**Figure 8-41** Details of an AS2 exchange with encryption and signatures
4. Select the AS2 document to open the Raw Document Viewer, shown in Figure 8-42. Again compare the content type in Figure 8-42 with the content type in Figure 8-31 on page 219. The content types are the same since the two document manipulation techniques are executed on top of each other. These techniques are executed in the following order:

a. Sign
b. Encrypt

Thus, the encryption hides the signature.
5. Select the MDN document in Figure 8-41 to view the raw document. Figure 8-43 clearly shows a multipart message, where one part is clear text. Another part has as content type, the value message/disposition-notification. The last part has as content type, the value application/pkcs7-signature.
Implementing WebSphere BI Connect Express on Windows

After finishing the base setup of a B2B exchange between two business partners, you now extend it with a third partner, who uses WebSphere BI Connect Express. This chapter focuses on the installation and initial configuration of the Express edition of WebSphere BI Connect.
9.1 Overview of the Express edition

WebSphere Business Integration Connect – Express is a lightweight, easy to use, cost-effective B2B connectivity tool. It leverages AS2 standards for transmitting documents securely over the Internet. It provides a browser interface to manage the connection with trading partners and to manage the exchange of documents between trading partners. The interface provides the same look and feel as the Advanced and Enterprise editions of WebSphere BI Connect.

Similar to the full versions, WebSphere BI Connect Express provides tools to analyze, track, and investigate all aspects of the B2B exchange. You can review information about queued documents that are pending transmission and acknowledgements, and view historical information about successfully sent or failed documents. You can also view, add, and update public certificates and private keys that are used for encryption and digital signatures.

Sending and receiving documents is performed over either HTTP or HTTPS transport for Secure Sockets Layer (SSL) server-based authentication. The documents can either be sent without any further packaging or with the industry standard packaging AS2.

Sending documents can be performed in two ways. You can use the console interface and manually send a file by uploading it through the browser. Or, alternatively, WebSphere BI Connect Express can monitor a directory and send any files that are dropped there by other applications. Files that are received by the server are also stored in a directory on the file system. That means that any integration between WebSphere BI Connect Express and other internal applications is via the file system. The full version of WebSphere BI Connect offered has more options, such as Java Message Service (JMS) and Web services.

A wide range of document formats is supported by WebSphere BI Connect Express. Business formats such as electronic data interchange (EDI) documents (ANSI X12 and EDIFACT) and RosettaNet service content are supported by WebSphere BI Connect. Custom XML or standardized XML formats are also supported. In addition to structured data, WebSphere BI Connect can also process binary files.

To make sure that document exchange occurs in optimal and secure circumstances, WebSphere BI Connect Express supports standard algorithms for encryption, digital signatures, and compression.
Figure 9-1 shows a schematic overview of the outbound flow. WebSphere BI Connect Express can monitor a folder on the file system for the arrival of new files that are generated by a back-end application. When the file arrival is noticed, it is copied within the folder structure managed by WebSphere BI Connect Express. The process engine picks it up from there to perform document manipulation and packaging before it is transmitted to the trading partner via the Internet. Sending files can also be performed via the Console, which allows a direct interaction with the process engine. The Console also interacts with the process engine to perform any real-time monitoring of the document flow between the trading partners.

The inbound flow is quite similar. However, the Console cannot be used to receive files directly. All inbound files are stored in the inbound folder on the file system.

Figure 9-1  Outbound flow within a WebSphere BI Connect Express environment
9.2 Software installation and configuration

This section discusses the installation of WebSphere BI Connect Express on a single machine (Figure 9-2). There are a few options to configure or customize during the installation. Chapter 10, “Extending the basic B2B exchange” on page 249, uses this installation of WebSphere BI Connect Express on host name wbicxprs to interact with Company E.

![Figure 9-2 Single machine implementation of WebSphere BI Connect Express](image-url)
The installation of WebSphere BI Connect Express is a simple process. When you insert the CD and the autorun feature is active, a Web page (Figure 9-3) is launched. From here, you can read the readme file or connect to the documentation on the Internet or start the installation of the software itself.

![WebSphere Business Integration Connect — Express](image)

**Choose one of the following:**

- View the readme file
- Read about WebSphere Business Integration Connect at ibm.com
- Install WebSphere Business Integration Connect - Express
- Exit

*Figure 9-3  Launching the installation of WebSphere BI Connect Express*

When you select the install option, depending on your browser settings, you may see a file download warning that you can ignore. Click **Open** to start the actual installation program (Figure 9-4).

![File Download](image)

*Figure 9-4  Possible warning issued by the browser*
After you see the welcome window, you are asked to confirm acceptance of the license. You also need to provide an installation folder, for example, C:\WBICConnect-Express.

In the following windows, you specify the following details:

- The port number for HTTP
- The folder name in the Start Programs menu
- The option to run WebSphere BI Connect Express as a service

After you specify this information, you see a summary window (Figure 9-5) that lists all your selections. Click **Next** to start the actual installation.

![Figure 9-5  Installation summary](image)
When the installation is complete, you are ready to start the server. You can do this by using the Windows Services application, if you choose to create a service during the installation. Or, you can start the server using the shortcut in the Start Programs menu. In this case, a text console (see Figure 9-6) shows some logging information that is refreshed automatically during normal operations.

**Attention:** Do not try to stop the server via this interface. Stopping the server is performed through its browser-based console.

![Start Server](Image)

**Figure 9-6** Online logging in console window

### 9.3 Initial configuration of the WebSphere BI Connect Express server

You manage the server (including stopping it) using a browser-based console. To open this console, a shortcut has been added to the Start Programs menu.

1. Select **Start** → **Programs** → **Business Integration Connect - Express** → **Console**.

**Note:** This shortcut refers to a Web link that includes a reference to the port number. If you set a non-standard port in the installation, you may have to update this shortcut to use the custom port number. The URL behind the shortcut is:

   http://localhost:80/qc/index.jsp
2. For the initial logon, use the user ID admin who has the password admin. Click Login (Figure 9-7).
3. During this first logon, you must change the password of the user ID admin. 
   You must also change the password of another built-in user ID, Guest. 
   Provide new passwords for both users and click **Save** (Figure 9-8).

![WebSphere Business Integration Connect - Express](image)

**Figure 9-8  Initializing passwords during initial logon**
4. After you set the new passwords, you return to the original main logon window. This time, logon using the user ID admin and the new password (Figure 9-9).

![Figure 9-9  Logon using the new password](image)

5. When the logon is complete, the Create Participant window (Figure 9-10) is displayed. Provide basic information about the first participant with whom the owner of this server of WebSphere BI Connect Express wants to communicate. Also specify the options or features that you want to use in that exchange.

Except for the Participant Name field (the first one), you can change all parameters later.

a. Provide a participant name, such as companyE.

b. Select the inbound protocols that you want to support: HTTP, HTTPS, or both.

c. If you want to activate user alerts, provide information about the e-mail server in your environment.
Figure 9-10  Create Participant (Part 1 of 2)
d. Scroll further down to the section labeled Capabilities (Figure 9-11). Here, you indicate the kind of documents that can be received.

You also provide the AS2 participant ID. You must provide this ID when you want to use AS2. It is used in the AS2 header information when sending documents to your partners.

You can transport many types of documents within an AS2 envelope. Some of these documents carry routing information and business identifiers, such as EDI. If you want to send or receive EDI documents, select the appropriate option to activate it and click **Save**.

![Capabilities](image)

**NOTES:**
- Participant ID is required if any AS2 capabilities are enabled.
- E-mail host and e-mail recipients are required if User Alerts are enabled.
- Binary Content Type is required if AS2 binary capabilities are enabled.

*Figure 9-11  Create Participant (Part 2 of 2)*

When the information entered in Figure 9-10 and Figure 9-11 is stored, the normal interface of WebSphere BI Connect Express becomes available. First, in the Manage Participants window, shown in Figure 9-12, you see that the configuration work is not yet finished. We complete this in Chapter 10, “Extending the basic B2B exchange” on page 249.

In the top-right corner, you see icon buttons that you can use to either stop or pause the server. The first three menu options, Reports, AS2, and HTTP, provide several options for you to learn about the documents that are sent and received, AS2 exchanges and their state and low-level HTTP information. The Configuration menu option provides you with all possible configuration settings,
again grouped in logical units: participants, AS2 and HTTP. The Security menu option brings together all settings related to security including the management of certificates.

The initial configuration of the server is now complete and verified.

Upon saving the participant’s profile, a number of directories are created on the file system. The directories are used to store both messages received from and sent to the participant. You can find the newly created directories in C:\WBIConnect-Express\data\FileSystemAdapter2\partners\companyE.
Extending the basic B2B exchange

This chapter continues the configuration of WebSphere BI Connect Express so that XML documents can be exchanged between Company E and Company X. It also discusses the changes that are required to the configuration of WebSphere BI Connect Enterprise server of Company E to support this additional partner.

Since the structure of the XML documents is not known to WebSphere BI Connect Enterprise, additional setup is required so that WebSphere BI Connect Enterprise can handle this new type of documents. This setup is required for routing purposes but can also be used for document validation.
10.1 Scenario overview

This chapter extends the base B2B environment by connecting the WebSphere BI Connect Express installation of Company X to the WebSphere BI Connect Enterprise installation of Company E. Company E and X exchange XML documents over AS2 (Figure 10-1).

The XML documents that are exchanged are not in any specific standard. Company A and E have agreed upon a custom XML format that allows them to exchange purchase orders via the Internet. In this chapter, you add this custom XML format to WebSphere BI Connect so that it can find routing information in the XML document. When an XML document arrives at the receiver component of WebSphere BI Connect of Company E, WebSphere BI Connect looks for routing information in the XML document so that it can decide what to do with the document. This look-up process happens for inbound and outbound documents.

Figure 10-1 shows a schematic overview of the B2B exchange with the three partners. It also shows two connectors into WebSphere BI Connect of Company E. This can give you the impression that two different inbound streams are necessary or will be used.
In Chapter 7, “Creating a basic B2B exchange” on page 121, we created an HTTP target to accept inbound electronic data interchange (EDI) documents from Company A. That object referred to the following URI:

http://wbichub:57080/bcgreceiver/companye/edi_in

Does this now imply that we have to create a separate target for inbound XML documents from Company X? We can, but there is no need for that. For any document arriving at that URI, WebSphere BI Connect tries to find a matching definition, based on that definition routing information.

This routing information is in the EDI document (ISA segment) and in the XML document. WebSphere BI Connect contains the definitions of EDI documents, but does not contain yet a definition for the custom XML document. By adding it to WebSphere BI Connect and by providing the XML elements of the custom XML document that contains routing information, WebSphere BI Connect can perform the same type of routing.

A similar discussion can be held about the file system gateway. Earlier we defined a file system gateway called \WBIConnect\data\companye\edi_in. This folder was used by WebSphere BI Connect to store any received EDI documents. Depending on other processing that has to occur for incoming documents, you may prefer to store incoming XML documents in a different directory. Maybe back-office applications are easier to configure if XML and EDI documents are stored in a different folder. WebSphere BI Connect can do it either way.

The same applies for the file system target. EDI documents were picked up by WebSphere BI Connect from the \WBIConnect\data\companye\edi_out directory. You can use this same folder again to send XML documents to Company X. Or, if you prefer, you can create a new file system target.
10.2 Implementation steps

To enable the exchange of XML documents between Company E and X, we need to finalize the configuration of WebSphere BI Connect Express of Company X and perform additional configuration of WebSphere BI Connect Enterprise of Company E.

On Company X’s server, we need to complete the profile of Company E and Company X and review the settings for HTTP and AS2. Since WebSphere BI Connect Express uses a different way of routing documents, it does not need to know the structure of the XML document. The outbound routing is performed based on where the document is found on the file system.

On Company E’s server, more work is needed. First, a profile is created for the new partner Company X. Next, a new document flow for the new custom XML format is defined. It is linked to the packaging models AS2, None, and Backend Integration. Then, an interaction is created for documents from None to AS2 packaging and vice versa. When the new document flow and interaction are created, we can update the B2B Capabilities of the partners Company E and Company X.

At this point, the participant connections are created and configured for sending and receiving custom XML documents over AS2 between Company E and Company X.

10.3 Configuration of the Company X machine

The tasks to perform by the administrator of WebSphere BI Connect Express can be divided into two categories. They need to update the profile of the participant Company E that was created immediately after the initial logon, and they need to update their own profile.
10.3.1 Customizing the profile of participant Company E

During the initial startup, we created a partial profile for Company E, as the partner of Company X who owns the WebSphere BI Connect Express server. We now proceed by completing that profile.

To finish the settings for HTTP, follow these steps:

1. Start the WebSphere BI Connect Express console. Select Start → Programs → Business Integration Connect - Express → Console.

2. Log on as admin user.

3. From the main menu of the console, select Configuration → HTTP.

4. In the Manage HTTP window (Figure 10-2), for Selected Participant, make sure that companyE is selected. Click Edit.

![Figure 10-2 HTTP settings for partner Company E](image-url)
5. In the next window (Figure 10-3), complete these tasks:

   a. For the inbound communication, provide a user name and password. This needs to be used by the server of Company E when setting up a profile to send to Company X. WebSphere BI Connect Express uses this incoming user name and password to identify and authenticate the sending server.

   b. For the outbound communication, provide the URL on the server of Company E that is configured as a target. You do not need a user name and password for outbound communication, since WebSphere BI Connect uses other techniques for identifying the partner that sent the incoming document. If Company E was using WebSphere BI Connect Express as well, then it would be required to provide a user name and password.

   c. Request an asynchronous message disposition notification (MDN) and click **Save** to store the updated HTTP settings for communication with Company E.

![WebSphere Business Integration Connect Express](image)

**Figure 10-3**  **Updating the HTTP settings related to partner Company E**

6. In addition to the HTTP settings, look at the settings for AS2. Select **Configuration → AS2**.
7. In the Manage AS2 window (Figure 10-4), for Selected Participant, select companyE. When you see the current settings, click **Edit**.

8. Update the destination address for AS2 over HTTP. Set it to the target that is defined on the server of Company E. For now, you can accept all defaults and click **Save**.

**Note:** For inbound AS2, the settings are copied from My Profile. Reviewing and updating My Profile is the next task.

![Figure 10-4  Editing the AS2 settings related to partner Company E](image)
10.3.2 Customizing My Profile

Next, in configuring WebSphere BI Connect Express for both sending and receiving XML over AS2, you must review and edit My Profile. My Profile in WebSphere BI Connect Express contains both the properties of the machine on which WebSphere BI Connect Express is installed and the business details that relate to the organization that owns the installation.

1. To edit, select **Configuration → My Profile**. Click **Edit** to change the current values.

2. At the minimum, provide the host name and port for incoming AS2 documents and the AS2 identifier, for example, companyx (see Figure 10-5). The complete URL for incoming AS2 documents is:

   http://wbicxprs/input/AS2

   For plain HTTP, WebSphere BI Connect Express listens on:
   http://wbicxprs/input/HTTP

   Click **Save** to store the edited profile.

---

**Figure 10-5  Managing My Profile for Company X**
Additional configuration for Company E

Given that WebSphere BI Connect Enterprise provides more features than WebSphere BI Connect Express, it comes as no surprise that the configuration of WebSphere BI Connect Enterprise includes a few more steps. Changes have to be made to the global profile of hubadmin and the profile of Company E. Also, a new profile for Company X has to be created.

10.4.1 Updating the profile of hubadmin

The hubadmin must create the new partner. He must also define a new protocol and document format.

Creating a new Community Participant

First, on the WebSphere BI Connect Enterprise hub, you must define a new Community Participant that identifies the WebSphere BI Connect Express server of Company X.

1. Log on as hubadmin to the console of WebSphere BI Connect.
2. Select Account Admin → Profiles → Community Participant. Click Create.
3. Use Table 10-1 as a guide to populate the fields in the New Participant window (Figure 10-6). Remember that Participant Login Name is case-sensitive and is used as the company name in the logon window. The Freeform Business ID is used for routing purposes.
   a. Provide a login (or company) name for the new company, for example, companyX.
   b. Provide a participant name, for example, Company X. This name is used in the interface.
   c. For Participant Type, select Community Participant.
### Table 10-1  Attributes to populate when creating a new participant

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant Login Name</td>
<td>companyX</td>
</tr>
<tr>
<td>Participant Name</td>
<td>Company X</td>
</tr>
<tr>
<td>Participant Type</td>
<td>Community Participant</td>
</tr>
<tr>
<td>Freeform Business ID</td>
<td>companyx</td>
</tr>
<tr>
<td>Gateway</td>
<td>wbicxprs</td>
</tr>
</tbody>
</table>

**Figure 10-6  Creating a new participant**
Creating a new document flow definition

WebSphere BI Connect does not include a document flow definition for custom XML formats. Given that these XML formats are essentially freeform, each custom XML format to be processed by WebSphere BI Connect must first be defined as a document flow definition.

The hubadmin user must perform this task.

1. To create a new document flow definition for our custom XML format, select **Hub Admin → Document Flow Definition**.

2. When the existing document flow definitions are shown, click **Create Document Flow Definition**.

3. The document flow definition that we create is at the Protocol Level, which means that this XML document is defined at the same level as an EDI standard, for example. Use the values in Table 10-2 for this new document flow definition (Figure 10-7).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document flow type</td>
<td>Protocol</td>
</tr>
<tr>
<td>Code</td>
<td>customXML</td>
</tr>
<tr>
<td>Name</td>
<td>customXML</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Document level</td>
<td>No</td>
</tr>
<tr>
<td>Status</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

a. Set Visibility to **Yes** for the:
   - Community Operator
   - Community Manager
   - Community Participant

b. Add this new protocol to:
   - Package: AS
   - Package: None
   - Package: Backend Integration

c. After you specify all values, click **Save**.
4. Create a document flow definition at the document flow level. Click **Manage Document Flow Definitions** or select **Hub Admin → Document Flow Definition** from the menu.

5. Click **Create Document Flow Definition**.
6. Use the values in Table 10-3 to complete the form shown in Figure 10-8 on page 262.

Table 10-3 Values to use when creating a document flow

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document flow type</td>
<td>Document Flow</td>
</tr>
<tr>
<td>Code</td>
<td>purchaseOrder</td>
</tr>
<tr>
<td>Name</td>
<td>purchaseOrder</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Document level</td>
<td>Yes</td>
</tr>
<tr>
<td>Status</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

a. Set Visibility to **Yes** for the three roles, as shown in Figure 10-8.
   
   We can compare the definition of this document flow to adding the definition of an 850 purchase order document to the EDI standard. If other XML documents need to be defined, you can add them directly at the document flow level. There is no need to create another document flow at the protocol level.

b. At the bottom of Figure 10-8, you see the expandable tree structure of packages and protocols. Expand this structure.

c. Add this new document flow to:
   
   - **Package: AS**
     → **Protocol: customXML**
   - **Package: None**
     → **Protocol: customXML**
   - **Package: Backend Integration**
     → **Protocol: customXML**
Figure 10-8 Creating a document flow at the document flow level
d. Figure 10-9 shows the completed structure. Click **Save**.

![Diagram of package and protocol structure]

**Figure 10-9  Link document flow to packages and protocols**

The next step is to create a new XML format and tie it to the document flow that we created.
Creating a new XML format

The main reason to define the XML format to WebSphere BI Connect is to make sure that the server knows how to route the XML document. It must use the data in the XML document to determine to which partner a document should be sent.

1. In the menu of the console of WebSphere BI Connect, while logged on as hubadmin, select **Hub Admin → Hub Configuration → XML Formats**.

2. The Manage XML Formats window (Figure 10-10) opens. Click **Create XML Format**.

![Figure 10-10 Managing XML formats in WebSphere BI Connect](image)

3. You see the View XML Format window (Figure 10-12). The new format needs to link to an existing protocol, which we defined earlier.
   a. For Routing Format, select **customXML 1.0**.
   b. For File Type, select **XML**.
   c. For Root Tag, select **Select Identifier**. The root tag in our XML documents is purchaseOrder. Another way to identify an XML document is to use a document type definition (DTD) or name space.
   d. In the Schema Attributes section, you tell WebSphere BI Connect how to identify the source and target business identifiers. These can be set as a constant. In most cases, you use an element in the XML document as the carrier for information about business identifiers. That way, it becomes easier to send this kind of XML documents to multiple partners. Figure 10-11 shows a sample XML document that indicates the element names that store business identifiers.
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE purchaseOrder>
<purchaseOrder>
  <fileHeader>
    <date>20030410</date>
    <time></time>
  </fileHeader>
  <orderHeader>
    <orderNumber>P345322</orderNumber>
    <orderFlag>0</orderFlag>
    <orderDate>20030410</orderDate>
    <contractNumber>49IR000123</contractNumber>
    <supplierId>companyx</supplierId>
    <supplierAddress1>Vimercate</supplierAddress1>
    <supplierAddress2>Italy</supplierAddress2>
    <supplierAddress3></supplierAddress3>
    <buyerName>Ronan Dalton</buyerName>
    <buyerSerial>i83731</buyerSerial>
    <currencyCode>USD</currencyCode>
    <paymentTerms>45</paymentTerms>
    <customerId>companye</customerId>
    <deliveryAddressName>IBM Internationl Holdings Ireland</deliveryAddressName>
    <deliveryAddress1>Damastown Industrial Estate</deliveryAddress1>
    <deliveryAddress2>Dublin</deliveryAddress2>
  </orderHeader>
  <lineDetail>
    <lineNumber>1</lineNumber>
    <partNumber>00021P0241</partNumber>
    <itemDesc>SCSI Card</itemDesc>
    <quantity>128</quantity>
    <uom>EA</uom>
    <unitPrice>98.62</unitPrice>
    <dueDate>20030630</dueDate>
  </lineDetail>
  ...
</lineDetail>
  <fileTrailer>
    <totalDollars>12623.36</totalDollars>
    <totalQuantity>128</totalQuantity>
  </fileTrailer>
</purchaseOrder>

Figure 10-11 Sample XML file (edited)
Figure 10-12 shows the form for creating a new XML format.

e. In addition to information about business identifiers, WebSphere BI Connect needs to know which document flow to invoke for XML documents of this format. This information can again be carried in the XML document itself. However, we make this information constant. The source document flow is purchaseOrder and the version is 1.0. The document flow was defined earlier (see Figure 10-8 on page 262).

f. Click **Save** to store the new format.

![Figure 10-12 Creating a new XML format](image)

### Creating an interaction

Earlier, we created the protocol customXML that can be packaged in AS2, None, or Backend Integration. Then we created the document flow purchaseOrder that
was linked to the protocol customXML for each packaging method. Next we defined the XML format to link to that document flow. Now we create the interactions that tell WebSphere BI Connect how to move from one packaging, protocol, or document to another.

1. While logged on as hubadmin, select **Hub Admin → Hub Configuration → Document Flow Definition**.

2. Click **Manage Interactions**.

3. Click **Create Interaction**.

4. We need an interaction from AS/customXML/purchaseOrder to None/customXML/purchaseOrder, which is the interaction shown in Figure 10-13. For Action, select **Pass Through** and click **Save**.

![Figure 10-13 Creating an interaction from packaging AS to None for customXML](image)
5. Repeat the same steps to create the reverse interaction, from None/customXML/purchaseOrder to AS/customXML/purchaseOrder. This interaction is shown in Figure 10-14. Again set Action to Pass Through and click Save.

Figure 10-14  Creating an interaction from packaging AS to None for customXML
10.4.2 Updating the profile of Company X on the Company E server

When we configured the AS2 communication between Company E and Company A in Chapter 7, “Creating a basic B2B exchange” on page 121, we described the configuration process in a role-based way. The hubadmin and the administrators of Company E and Company A had their own tasks to complete. Remember that as an administrator of a company, you have to perform tasks on your own server and on the server of your partner.

In 7.8, “Revisiting role-based configuration” on page 184, we explained that the role-based approach is not required. For example, companies may not feel comfortable about the fact that their partners log on directly to their systems. Another reason why you may not want to follow the role-based approach is the use of different software products by your partner. You may not want to educate your partners about WebSphere BI Connect if they use a different AS2 provider.

This is why we do not follow the role-based approach in this section. Company X uses the Express edition of WebSphere BI Connect. It is not familiar with the console interface of the Advanced or Enterprise editions of WebSphere BI Connect. Therefore, the hubadmin updates the profile of Company X on behalf of the administrator of Company X.
1. While logged on as hubadmin, select **Account Admin** → **Profiles** → **Community Participant**.

2. In the Participant Search window (Figure 10-15), click **Search**. There is no need to provide a search argument since only four profiles are defined. When you see the list of profiles, click the icon next to Company X.

![Participant Search Image](image)

**Figure 10-15**  List of participants at Company E

3. Set the B2B capabilities of partner Company X as shown in Figure 10-16.

   a. Select **B2B Capabilities**.
   b. Verify that you are working with the profile of Company X.
   c. Enable AS, customXML, and purchaseOrder as the source and target.
   d. Enable None, customXML, and purchaseOrder for the source and target.
4. Create the HTTP gateway that points to the WebSphere BI Connect Express server of Company X. While working with the profile of Company X, select Gateways.

5. When you see the list of gateways, click Create.

6. Use the values in Table 10-4 to complete the form in the Gateway List window (Figure 10-17). This time, the attributes User Name and Password need values, because Company X hosts WebSphere BI Connect Express. These attributes were not used when we created a gateway for Company A, for example. We created the user name companye and a password during the profile customization of Company E on the WebSphere BI Connect Express server of Company X. This user name is not defined on the server of Company E and is not related to business identifiers.
Table 10-4  Creating the default gateway for Company X

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway Name</td>
<td>HTTPGateway</td>
</tr>
<tr>
<td>Transport</td>
<td>HTTP/1.1</td>
</tr>
<tr>
<td>Target URI</td>
<td><a href="http://wbicxprs/input/AS2">http://wbicxprs/input/AS2</a></td>
</tr>
<tr>
<td>User Name</td>
<td>companye</td>
</tr>
<tr>
<td>Password</td>
<td>partner_e</td>
</tr>
</tbody>
</table>

Make sure to set the status to *Enabled*. Click **Save** to store the new gateway.

![Image](image_url)

Figure 10-17  Creating a new gateway
7. When the gateway is created, set it as the default gateway for Company X. Return to the list of gateways for Company X and click **View Default Gateways**.

8. For the production gateway, select **HTTPGateway** and click **Save**. You should see the list of gateways for Company X as shown in Figure 10-18.

![Figure 10-18](image)

**Figure 10-18**  List of gateways for Company X

### 10.4.3 Updating the profile of Company E on the Company E server

In Chapter 7, “Creating a basic B2B exchange” on page 121, we set the B2B capabilities of Company E so that it could send and receive EDI documents over AS2. Now we must update these B2B capabilities to include custom XML documents.

1. While logged on as hubadmin, open the profile for Company E.
2. Select **B2B Capabilities**.
3. In the B2B Capabilities window (Figure 10-19), update the capabilities by enabling AS, customXML, and purchaseOrder, as well as None, customXML, and purchaseOrder.
There is no need to define another gateway. While the existing gateway refers to edi_out in its URI, there is no need to create a specific gateway for XML. Now you must create new participant connections, the final step in the configuration.

1. While logged in as hubadmin, select **Account Admin → Participant Connections**.
2. Select **Company E** as the source and **Company X** as the target. Click **Search**.
3. Locate the connection that takes an unpackaged XML document and sends an AS2 packaged document. See Figure 10-20. Click **Activate**.

![Figure 10-20  Connection to send AS2-XML to Company X](image)

4. Click **Attributes** for this connection to set AS2 attributes in the same way as done before.

5. Provide an e-mail address and an HTTP address for MDNs, for example:
   - MDN E-mail address: mdn@companye.com
   - MDN HTTP URL: http://wbichub:57080/bcgreceiver/companye/edi_in

This completes the configuration required on the WebSphere BI Connect Enterprise server to allow for the sending of a custom XML format over AS2 to a WebSphere BI Connect Express server.

Before you send a sample XML file over AS2 from the WebSphere BI Connect Express server to the WebSphere BI Connect Enterprise server, you must activate another connection.

Return to the search window for participant connections. This time, set **Company X** as the source and **Company E** as the target. Click **Activate** for the connection that takes an AS packaged document and turns it to no-packaging (see Figure 10-21).

![Figure 10-21  Connection to receive AS2-XML documents from Company X](image)
10.5 Validating communication

Now you are ready to validate the configuration by sending XML documents between both companies.

10.5.1 Sending XML documents from Company E to Company X

All should be in place now. To test the configuration, again you can use the test connection feature first. Assuming that this feature does not report an error, it is sufficient to drop an XML file in the edi_out folder used previously to send EDI documents to Company A. Again WebSphere BI Connect can determine that the document is an XML document. There is no need to sort any documents in format-specific directories or partner-specific directories.

Compared to the processing that happened between Company E and Company A, there are a few differences. Having picked up the file and wrapped for AS2, WebSphere BI Connect Enterprise must first establish an HTTP connection with WebSphere BI Connect Express. This connection is established only on the presentation of a valid user name and password by the WebSphere BI Connect Enterprise server. In essence, the WebSphere BI Connect Enterprise hub is logging into the servlet employed by WebSphere BI Connect Express for receipt of HTTP POSTs.

When this connection is established, WebSphere BI Connect Express knows the source of the message. The user name and password should match the detail provided when creating the HTTP configuration for the participant in WebSphere BI Connect Express.

The WebSphere BI Connect Express server determines whether packaging is to be stripped by the URI on which it receives. Messages sent to a URI with the extension /input/HTTP are received and placed in the file system in the folder HTTP. Messages sent to a URI with the extension input/AS2 are assumed to have AS2 MIME headers. These messages are stripped of any AS2 headers. The payload, in our case an XML file, is placed in the file system in the folder AS2. The complete directory is called C:\WBIConnect-Express\data\FileSystemAdapter2\partners\companyE\received\AS2\XML.

Besides looking at the destination folder, you can use the AS2 Viewer on the server of Company E to learn whether the transmission succeeded. Or, you can use the viewers that are available in WebSphere BI Connect Express.

1. Log into the console of WebSphere BI Connect Express as the admin user.
2. From the menu, select **AS2 → Received**.
3. Enter any search parameters and click **Search**. You see a list of received documents that includes some statistical information, as shown in Figure 10-22.

Similar to the AS2 Viewer for WebSphere BI Connect Enterprise, you can obtain more details by selecting the eye-glass icon, and look at the actual document.

![AS2 Received Documents Viewer of WebSphere BI Connect Express](image)

**Figure 10-22** AS2 Received Documents Viewer of WebSphere BI Connect Express

### 10.5.2 Sending XML documents from Company X to Company E

You can send messages from the WebSphere BI Connect Express server by either using the WebSphere BI Connect Express console or by dropping a file in the right directory.
Manual sending
For manual sending, you use these steps:

1. While logged into the console as the admin user, select **AS2 → Send**.

2. Select **companyE** as the participant. Set the content type to **XML**. Click **Browse** to locate the file being sent on the file system. Click **Send** to initiate the transmission (Figure 10-23).

The file is wrapped with AS2 MIME headers and sent to Company E over HTTP. You can see the message in the AS1/AS2 Viewer of Company E as before.

![WebSphere Business Integration Connect Express](image)

**Figure 10-23** Manually sending an XML document

3. To verify the transmission, use the AS2 Viewer of WebSphere BI Connect Express.
   a. Select **AS2 → Sent**.
   b. Provide any search arguments.
   c. Click **Search**.

You see a list of sent documents and some statistical information as shown in Figure 10-24.
Automatic sending

The second means of sending files from the WebSphere BI Connect Express server is to use the file system. Similar to WebSphere BI Connect Enterprise, WebSphere BI Connect Express can read files from the file system and send to participants over HTTP or HTTP using AS2. Unlike WebSphere BI Connect Enterprise, it is not possible to define from which directory WebSphere BI Connect Express reads the files to be sent.

For each partner, for each protocol, and for each document format, there is a directory from which files are read by WebSphere BI Connect Express. For XML documents sent over AS2 to Company E, the name of the directory is C:\WBICConnect-Express\data\FileSystemAdapter2\partners\companyE\send\AS2\XML. Similar directories exist for HTTP and for document formats such as EDI.

To send a message to Company E in this manner, simply copy an XML file in this location. Via this directory, WebSphere BI Connect Express can be integrated with any back-end application systems.
This chapter discusses another extension of the B2B solution of Company E. Company F and E want to establish a B2B exchange. However, Company F is not ready to implement AS2. It prefers a FTP-based solution.

WebSphere BI Connect provides support for sending documents over FTP to the FTP server of Company F. To receive documents from Company F over FTP, you must integrate the FTP server of Company E into the WebSphere BI Connect server of Company E.
11.1 Scenario overview

In a typical B2B environment, it is common that not all business partners can be aligned along a single communication mechanism. While AS2-based solutions are common and not that difficult to implement, there are always situations where a business partner cannot implement an AS2 solution and for many reasons. However, a B2B software product, such as WebSphere BI Connect, can handle these situations.

This chapter considers the situation where Company E wants to extend its B2B infrastructure to include a document exchange with Company F. However, Company F is reluctant to jump quickly on a technology that is new to the IT department of Company F. At the same time, Company E wants to build a solution sooner rather than later. A compromise is found between Company E and Company F by implementing an FTP-based solution. Company F has extensive experience with FTP and can implement a document exchange with Company E rather quickly.

Figure 11-1 shows a schematical overview of the complete B2B solution from the perspective of Company E. It shows the existing AS2-based exchanges with Company A (for electronic data interchange (EDI)) and with Company X (for XML). Within this infrastructure, we explain how to add an exchange with Company F for FTP.

This solution requires FTP client and server support for both companies. When Company E wants to send a document over FTP to Company F, an FTP client on Company E is required. This FTP client contacts the FTP server of Company F to initiate the exchange. In the same way, an FTP client function is required at Company F, so that it can initiate an exchange with Company E. From an FTP perspective, we always use the put command to move a file from one company to another, and never the get command.

FTP client functionality is built into WebSphere BI Connect. However FTP server functionality is not. This chapter describes the use of the FTP product WFTPD and how you can integrate it with WebSphere BI Connect. The discussion is the same for any other FTP server.
In theory, it is possible to have a WebSphere BI Connect hub next to an FTP hub. However, this would mean that Company E loses a single point of control for all of its B2B transactions. By integrating the FTP server into WebSphere BI Connect, you can, for example, use the document viewer for all transactions with all trading partners. Also, whenever Company F is ready to upgrade its FTP server to a full B2B hub, Company E does not have to make huge changes to the routing logic in its back-end applications or to their B2B hub. Adjusting the gateway to Company F from an FTP gateway to an HTTP gateway, for example, is an easy change.

### 11.2 Configuration of Company E for outbound

This section discusses the outbound flow. First you learn how the FTP client feature of WebSphere BI Connect works. Then you see the actual implementation of such a solution.
11.2.1 Implementation steps

When sending binary files, WebSphere BI Connect cannot inspect the actual document to learn about the sending and receiving business identifiers. For custom XML, we defined the XML format to WebSphere BI Connect so that it can retrieve routing information. For EDI-X12, this knowledge is built into the product itself. But, for binary documents, this document inspection technique is not available.

Another typical location where routing information can be stored or retrieved is the package layer. For example, when using AS2, WebSphere BI Connect can use the AS2 identifiers to perform routing tasks. However, when using FTP, there is no package layer that can be used.

To make sure that documents are routed to the correct destination, WebSphere BI Connect uses a specific receiver, called FTP Directory. This receiver should be used when sending documents over FTP to your partners. This receiver tells WebSphere BI Connect what the intended sender ID and receiver ID are, based on the location (folder name) where the to-be-sent file is found, and the name of the actual file that needs to be sent.

For example, if Company E wants to send a file via the FTP client of WebSphere BI Connect, then the back-end applications of Company E need to write this file in the folder companyE within the root directory that is configured for the receiver. The name companyE is the login name of Company E within the WebSphere BI Connect Enterprise server of Company E (see Figure 7-16 on page 140). The fact that the file is a binary file is derived from the name of the folder as well. Also the type of transmission, production or test, is derived from the folder name. The complete folder structure can be \WBIConnect\data\ftp\companyE\Binary\Production. In this case, \WBIConnect\data\ftp is the name of the folder that is used in the configuration of the FTP Directory receiver.

The receiver ID is derived from the file name. Thus, the back-end application of Company E needs to create a file in the folder companyE with a name that starts with companyF. The value companyF is the business identifier of Company F. The folder name is named after the login ID, and the file name is named after the business ID. A sample file name can be companyF.1234567890.bin.

When the receiver retrieves the file and passes it to the document manager, an interaction is required that takes a binary unpackaged document to a binary unpackaged document. Thus, you must ensure that B2B capabilities are enabled for None/Binary for Company E and Company F. You must also create a participant connection between Company E and Company F for this interaction. Finally, you need an FTP gateway for Company F that points to the FTP server of Company F.
Figure 11-2 summarizes the configuration points for this scenario:

1. Create a root-level folder that is used by the FTP Directory receiver.
2. Create a folder structure that starts with companyE.
3. Create the target FTPTarget of transport type FTP Directory.
4. Create a partner profile for Company F with business ID companyf.
5. Update the B2B capabilities of Company F to include None/Binary as source and target.
6. Update the B2B capabilities of Company E to include None/Binary as source and target.
7. Create an interaction between None/Binary and None/Binary for the action Pass-Through.
8. Create a gateway of type FTP within the profile of Company F.
9. Create a participant connection that brings together the interaction and the partners.
11.2.2 Creating a directory structure

Using Windows Explorer, follow these steps:

1. Create the directory that acts as the root directory for all outbound FTP communications via WebSphere BI Connect. In our setup, we create the directory ftp within the existing directory WBIConnect\data.

2. Within this folder ftp, create a new folder called companyE, which is the login ID of Company E. The name of this folder is case-sensitive.

3. Within the folder companyE, create two more folders: Binary and Documents.

4. For these two folders, create two more subdirectories: Production and Test.

Figure 11-3 shows the complete directory structure that is required to send binary and XML documents via FTP by Company E.

![Directory Structure Screenshot](Image)

Figure 11-3 Required directory structure for outbound FTP
11.2.3 Updating the profile of hubadmin

As the hub operator, you must perform the following tasks:

1. Create a partner profile for Company F.
2. Create a new target for the FTP Directory receiver.
3. Create an interaction from unpackaged binary (None/Binary) to unpackaged binary (None/Binary).

Creating a new Community Participant

To create the new partner profile for Company F, complete these steps:

1. Log on as hubadmin to the WebSphere BI Connect server of Company E.
2. Select Account Admin → Profiles → Community Participant.
3. Click Create.
4. In the New Participant window (Figure 11-4), do these tasks:
   a. For Participant Login Name, type companyF.
   b. For Participant Name, type Company F. This is consistent with the profiles that we created earlier for Company A and Company X.
   c. For Participant Type, select Community Participant.
   d. Under Business ID, for Type, select Freeform and for Identifier, select companyF.
   e. Under IP Address of Host Name, add a gateway.
   f. Click Save.

Note: The value of the business ID is linked to the name of the files that we want to send. It cannot be any random value.
Figure 11-4 Creating a participant profile for Company F

**Creating a new target**

A new target is required to retrieve the files for Company F into WebSphere BI Connect and to send them via FTP to Company F. To enable the integration with FTP, a specific type of receiver is required.

1. Select **Hub Admin → Hub Configuration → Targets**.
2. When the list of currently defined targets is shown, click **Create Target**.
3. In the Target Details window (Figure 11-5), complete these tasks:
   a. Provide a name for the new target, for example FTPTarget.
   b. Set the transport to FTP Directory. This displays more attributes that are specific for this kind of transport.
   c. Set FTP Root Directory to \WBIConnect\data\ftp, which is the folder that we created in 11.2.2, “Creating a directory structure” on page 286.
   d. All other attributes can have default values.
   e. Click Save.

**Note:** You can configure the receiver to ignore files with a certain extension. We do not need this feature in our setup.

![Figure 11-5 Creating a new target of type FTP Directory](image-url)
Creating an interaction
The required interaction is likely the most simple interaction that can be configured within WebSphere BI Connect. Nevertheless, you must enable it before it can work.

1. While logged on as the hubadmin, select Hub Admin → Hub Configuration → Document Flow Definitions.

2. In the Management Document Flow Definitions window (Figure 11-6), verify that the combinations None/Binary and None/customXML/purchaseOrder are enabled.

Note: Refer to Part 4, “Extending the features of WebSphere BI Connect” on page 487, which discusses the appearance of the ITSO_XML protocol and ITSO package in the list of document flow definitions and later in the list of B2B capabilities.
3. To review the interactions, select **Manage Interactions**.

4. Click **Create Interaction**.

5. In the Manage Interactions window (Figure 11-7), complete these steps:
   a. Expand the tree structures under the Source and Target headings.
   b. Locate **None** and **Binary** as the source and target.
   c. For Action, select **Pass Through**.
   d. Click **Save**.

---

**Figure 11-7**  Interaction for None/Binary to None/Binary
6. As shown in Figure 11-8, complete these tasks:
   
a. Create an interaction between None/customXML/purchaseOrder as the source and None/customXML/purchaseOrder as the target.

   b. For Action, select **Pass Through**.

   c. Click **Save**.

![Manage Interactions](image)

Figure 11-8 Interaction for None/customXML/purchaseOrder for source and target

11.2.4 **Updating the profile of Company F**

In the previous section, we created a profile for Company F. This profile needs further configuration. You need to perform the following tasks:

1. Set the B2B capabilities of Company F.
2. Create the FTP gateway that points to the FTP server of Company F.
Follow these steps:

1. While logged on as the hub administrator, select **Account Admin → Profiles → Community Participant**.

2. Click **Search**.

3. Locate the profile of Company F and click the magnifying glass icon next to it. Select **B2B Capabilities** to indicate that Company F can handle binary files and custom XML files that have no packaging.

4. In the B2B Capabilities window (Figure 11-9), enable customXML and Binary for packaging None as the source and target.

![Figure 11-9 Setting B2B capabilities for Company F](image-url)
5. Create the FTP gateway. While working with the profile of Company F, select **Gateways**.

6. You see an empty list of gateways. Click **Create**.

7. In the Gateway List window (Figure 11-10), complete these items:
   a. For Gateway Name, type **FTPGateway**, for example.
   b. For Transport, select **FTP**.
   c. For Target URI, type **ftp://wbicftp**. The value **wbicftp** is the host name of the FTP server of Company F.
   d. If the FTP server is protected by a user ID and password, provide these values as well.
   e. All other attributes can have default values.
   f. Click **Save**.

![Figure 11-10 Creating the FTP gateway within the profile of Company F](image)
8. Each partner profile needs a default gateway. Return to the list of gateways for Company F and click **View Default Gateways**.

9. Select **FTPGateway** as the production gateway and click **Save**. You should now see the list of gateways for Company F as shown in Figure 11-10.

![List of gateways for Company F](image-url)
11.2.5 Updating the profile of Company E

The changes to the profile of Company E are limited to reviewing the B2B capabilities of Company E.

1. Select **Account Admin → Profiles → Community Participant**.
2. Click **Search** and click the magnifying glass icon next to Company E.
3. While working with the profile of Company E, select **B2B Capabilities**.
4. In the B2B Capabilities window (Figure 11-12), enable None/Binary for the source and target. The combination None/customXML/purchaseOrder should already be enabled as part of the configuration performed in Chapter 10, “Extending the basic B2B exchange” on page 249. Figure 11-12 shows the B2B capabilities of Company E.

![Figure 11-12   Updating the B2B capabilities of Company E](image)
11.2.6 Creating a participant connection

The last step in the configuration of the WebSphere BI Connect server of Company E is to activate the required participant connections.

1. While logged on as the hub administrator, select Account Admin → Participant Connections.

2. Select Company E as the source and Company F as the target. Click Search.

3. You now see several possible connections, based on the common B2B capabilities of Company E and Company F. The required connections for these two companies are:
   - From None/Binary to None/Binary
   - From None/customXML/purchaseOrder to None/customXML/purchaseOrder

Click Activate for these two connections.

![WebSphere Business Integration Connect Community Console](image)

Figure 11-13  List of possible connections between Company E and Company F
Figure 11-14 shows the resulting list of connections. You can click **Gateways** to review the gateway selection. However, by default, FTPGateway should be the only defined gateway in the profile of Company F.

**11.2.7 Validating communication**

The configuration of the FTP server of Company F, or even its nature, is not that important for the scope of this redbook. It is sufficient to say that Company F has an FTP server and that the user ID company_e is defined and authorized to access this FTP server. If you are interested in recreating the setup used for the development of this redbook, review 11.3.1, “Installing an FTP server” on page 302, and 11.3.2, “Configuring the FTP server” on page 303. Those sections discuss the setup of the FTP server of Company E, which is needed to support inbound FTP. Company F in our lab environment uses a similar setup as its FTP server.
To validate the transfer of binary files, copy a file whose name starts with companyf (for example companyf.1234567890.bin) in the directory \WBIConnect\data\ftp\companyE\Binary\Production. At the next polling cycle, this file should be picked up by the target FTPTarget and handed to the document manager.

The document manager should then use its built-in FTP client function to send the file to Company F. You can again use the document viewer to monitor the progress and the result of the communication.

Figure 11-15 shows a successful exchange between Company E and Company F for binary files. Click the magnifying glass icon to view any events associated with this exchange.
Figure 11-16 shows all events associated with this document exchange. As you can see, the first delivery failed because the FTP server was intentionally shut down. After starting it, the WebSphere BI Connect server was able to write the file to the FTP server of Company F.

<table>
<thead>
<tr>
<th>Doc Time Stamp</th>
<th>Gateway Type</th>
<th>Connection Document Flow</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>In Time Stamp</th>
<th>Source Business ID</th>
<th>Source Document Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company F</td>
<td>12/16/04 0:50:00 PM</td>
<td>companyf</td>
<td>Binary 1.0: Binary 1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target</th>
<th>End State Time Stamp</th>
<th>Target Business ID</th>
<th>Target Document Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company F</td>
<td>12/16/04 0:55:17 PM</td>
<td>companyf</td>
<td>Binary 1.0: Binary 1.0</td>
</tr>
</tbody>
</table>

Figure 11-16  Events associated with the FTP exchange
Figure 11-17 shows the home directory of the user company_e. As you can see, the file was uploaded successfully. However, its name was altered to make sure that any uploaded file has a unique name. To preserve the file name, you can alter the configuration of the FTP gateway on the WebSphere BI Connect server of Company E. Figure 11-10 on page 294 shows the settings of this gateway. It also shows that the gateway was configured to use unique file names. You can turn off this option to preserve the file names.

In the same way, you can exchange XML documents. You create an XML file, for example companyf.1234567890.xml, that adheres to the structure defined to WebSphere BI Connect. Remember that the contents of this file are used by WebSphere BI Connect. This time, the XML element supplierId should be companyf, while the XML element customerId should be companye. You write such a file in the \WBIConnect\data\ftp\companyE\Documents\Production folder. The file should again be uploaded successfully to the FTP server of Company F.

### 11.3 Configuration of Company E for inbound

This section explains how to enable outbound FTP communication. It discusses how to integrate an FTP server with WebSphere BI Connect so that you can continue to use the features of WebSphere BI Connect while using an FTP server.
11.3.1 Installing an FTP server

As discussed earlier, WebSphere BI Connect itself does not provide a built-in FTP server. But, it is perfectly possible to integrate any FTP server with the product. For the purposes of this scenario, we use WFTPD Pro, which you can download from:

http://www.wftpd.com/

The installation of this FTP server is pretty straightforward. You can accept all default values during the installation.

![WFTPD Pro Setup - Welcome to WFTPD Pro](image)

Welcome to the installation for WFTPD Pro

Thank you for choosing our program - please pay careful attention to the "read me" text file, and also visit our help files and web site www.wftpd.com for more details.

Don't forget that if you have any questions about this program, you may contact us using the methods outlined in the "read me" text file, as well as in the help file.

Figure 11-18 Installing FTP server
11.3.2 Configuring the FTP server

The installation program of WFTPD Pro adds an icon to the standard Control Panel of Windows. You double-click the icon **WFTPD FTP Server** in the Control Panel to start the configuration and operational interface of WFTPD (see Figure 11-19).

To configure the server, you must perform the following three tasks:

1. Create a directory structure, such as a root directory for the server and a home directory for the users of the server.
2. Create a server instance.
3. Create a user ID for incoming connections.

![Figure 11-19 Control panel for the FTP server](image)

Creating a directory structure

For the outbound FTP communication, we use `C:\WBIC\Connect\data\ftp` as the root directory for the FTP Directory receiver. We can use the same directory as the root directory for the FTP server. Within this directory, create a new folder for Company F. As before, the name of this folder should match with the login name of Company F to the WebSphere BI Connect of Company E. This login name was set to companyF. See Figure 11-4 on page 288.
Within this folder companyF, create the same structure as before. This includes a subfolder named Binary, to receive binary files from Company F, and a subfolder named Documents, to receive XML documents. For each document type, create a subfolder named Production and Test, to separate the production and test document exchanges.

Figure 11-20 shows the complete directory structure.

![Directory structure to support inbound FTP](image)

**Creating a server instance**

The WFTPD FTP product supports several FTP server instances on a single machine. Each instance can listen on a different port for example. To create a new instance, follow these steps:
1. Click the **Create Server** icon button (circled in the toolbar) in Figure 11-22.
2. The Create Server window (Figure 11-21) opens. Provide a name for the server, such as `wbichub`, and click **OK**.

![Figure 11-21 Creating a new server](image)

3. After you return to the main interface, double-click the new server to configure more properties of it (see Figure 11-22).

![Figure 11-22 Selecting the new server to configure more properties](image)
4. In the Configure Server window (Figure 11-23), follow these steps:
   a. Select the **Security** tab.
      i. Select **Enable Security**.
      ii. Select the **Allow users to write to files** option.

![Configure Server Settings](image)

*Figure 11-23 Specifying the server-wide security settings*
b. Select the **Logging** tab (see Figure 11-24).
   
i. Select **Enable Logging**.
   
ii. In the Log to file field, provide a log file, for example C:\WBICconnect\data\ftp\wbicftp_log.ftp.
   
iii. Select the desired events for which you want to have a log record.

![Configure Server wbichub](image)

*Figure 11-24  Setting the server logging parameters*

c. Click **OK** to close the configuration dialog.
Creating a user ID for incoming connections

While it is common to have anonymous access to FTP servers, it is not something that we want to use in our environment. Every business partner who wants to upload files to the server, should have his own user ID, password and home directory. As such, we can be sure that Company F cannot obtain files that have been uploaded by other trading partners.

1. Click the **Configure Users** button in the main window of the WFTPD control panel.

2. A new window opens that allows you to define users and set access rights. Click the **New User** button (circled in Figure 11-25).

![Figure 11-25  User configuration window](image)

3. As shown in Figure 11-26, provide a user name, for example **company_f**. Company F will use this user ID when uploading files to Company E. Click **OK**.

![Figure 11-26  Creating a new user](image)
4. Enter a password for this new user name and click **OK** (see Figure 11-27).

![Set user password](image1)

*Figure 11-27  Setting a password for the new user*

5. Back in the User Configuration Dialog window (Figure 11-28), expand the folder structure in the right pane and locate the C:\WBIConnect\data\ftp\companyF folder. Click the **Set Home Directory** button (circled in Figure 11-29).

![User Configuration Dialog](image2)

*Figure 11-28  Configuring the home directory for user company_f*
6. To make sure that the user company_f can write to his home directory, deselect the **Inherit From Parent** option. Then set the access rights, as shown in Figure 11-29. The user company_f is now configured. Click **OK**.

*Figure 11-29  Setting access rights for user company_f*
The server is now fully configured. In the main control panel, restart the FTP service and start the server that you just configured in this section. When the service and the server are started successfully, you see a message indicating the success in the right pane, as shown in Figure 11-30.

![Figure 11-30  Restarting the server](image)

**Figure 11-30  Restarting the server**
11.3.3 Updating the configuration of WebSphere BI Connect

As part of the outbound document flow, we configured the B2B capabilities of Company E and Company F so that they can send and receive Binary and customXML documents that are not packaged. The only missing piece for the inbound flow is the participant connection.

1. Log on to the console of WebSphere BI Connect as hubadmin and select Account Admin → Participant Connections.

2. Select Company F as the source and Company E as the target. Click Search.

3. You see several possible participant connections as shown in Figure 11-31.
   a. Click Activate for the connection between None/Binary and None/Binary.
   b. Click Activate for the connection between None/customXML/purchaseOrder and None/customXML/purchaseOrder.

---

**Figure 11-31  List of possible participant connections**
You see the result in Figure 11-32.

Figure 11-32   List of activated connections between Company F and Company E

11.3.4 Validating communication

To validate the setup for inbound FTP, you need to send a file, by using FTP, from Company F to Company E. As before, the file name has to adhere to certain naming conventions. The file name should start with companye, which is the business identifier of Company E. The file name should also contain a portion that is unique. An example file name is companye.1234567890.bin. You send such a file by FTP to the server of Company E and upload it to the correct directory, which is \Binary\Production, within the home directory of the user company_f.
Figure 11-33 shows a portion of the dialog between the standard Windows FTP client and the FTP server hosted by Company E.

```
230 Logged in successfully
ftp> cd Binary
250 "/Binary" is current directory
ftp> cd production
250 "/Binary/production" is current directory
ftp> bin
200 Type is Image (Binary)
ftp> prompt off
Interactive mode Off.
ftp> mput companye*
200 PORT command okay
150 "/Binary/production/companye.1234567890.bin" file ready to receive in
IMAGE
/ Binary mode
226 Transfer finished successfully.
ftp: 40960 bytes sent in 0.00 Seconds 40960000.00 Kbytes/sec.
ftp>
```

*Figure 11-33  FTP client uploading a binary file to Company E*

After uploading the file, it is stored in the C:\WBIC\data\ftp\companyF\Binary\Production directory. The arrival of this new file is noticed by the FTP Directory receiver at its next polling cycle. The receiver passes this file to the document manager. The document manager has little work to do, since there is no unpackaging to be done. It stores the file in the configured default gateway for Company E, which is the FileSystemGateway. This is the same gateway that we have used many times throughout this redbook. It maps to the directory C:\WBIC\data\companye\edi_in. Once more, this demonstrates that WebSphere BI Connect can write any kind of document to a gateway. It also demonstrates that there is no need to name or create gateways for a specific document type or source.

Inspect the directory contents of C:\WBIC\data\companye\edi_in and you should see a new file with the name companye.1234567890.bin1103564740391000C29B10BBB001416000000000000006.

Notice again that the FTP Directory receiver has added a unique string to the file name. However, it is possible to preserve the original file name, if desired.
Another way to verify the successful processing of the file, that was received by the FTP server of Company E, is the use of the Document Viewer. Figure 11-34 shows the details of this document exchange. The source and target business identifiers have been recognized correctly.

![Document Details](image)

**Document Details**

File Name: companye.1234567890.bin  
Reference Id: 11035647612315008296100800001454000000000000000002  
Document Id: companye.1234567890.bin

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>companyf</td>
<td>Binary 1.0: Binary 1.0</td>
<td></td>
<td>companye</td>
<td>Binary 1.0: Binary 1.0</td>
</tr>
</tbody>
</table>

![Document Events](image)

**Document Events**

Event Filter:  
- Debug  
- Information  
- Warning  
- Error  
- Critical

Total Event Count: 4

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Time Stamp</th>
<th>Type</th>
<th>Event Code</th>
<th>Location</th>
<th>Source IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Parse Successful</td>
<td>12/20/04 5:45:30 PM</td>
<td>Info</td>
<td>BCG210040</td>
<td>Unknown</td>
<td>9.42.170.85</td>
</tr>
<tr>
<td>Destination Process Successful</td>
<td>12/20/04 5:45:30 PM</td>
<td>Info</td>
<td>BCG210052</td>
<td>Unknown</td>
<td>9.42.170.85</td>
</tr>
<tr>
<td>Document Sent to Outbound Processor</td>
<td>12/20/04 5:45:41 PM</td>
<td>Info</td>
<td>BCG210005</td>
<td>Unknown</td>
<td>9.42.170.85</td>
</tr>
<tr>
<td>Document Delivered</td>
<td>12/20/04 5:45:47 PM</td>
<td>Info</td>
<td>BCG250004</td>
<td>Unknown</td>
<td>9.42.170.85</td>
</tr>
</tbody>
</table>

*Figure 11-34  Using the document viewer for files received via FTP*

In the same way, we can exchange XML documents. Remember that the contents of this file are used by WebSphere BI Connect. This time, the XML element supplierId should be companye, while the XML element customerId should be companyf. You send such a file, by using FTP, to the FTP server of Company E. Then write it in the \Documents\Production folder within the home directory of the user company_f.

When the FTP Directory receiver finds the file, it posts it to the document manager who passes it on to the default gateway for Company E. Thus, this XML file is stored by WebSphere BI Connect in the C:WBICConnect\data\companye\edi_in folder.
You can again use the Document Viewer to inspect the processing of the incoming XML document via the FTP server. Figure 11-35 shows the details of such an exchange.

![Document Details](image)

So far, the main advantage of using WebSphere BI Connect in combination with the FTP server is that you have a single view of all types of document exchanges. Independent of protocol or document type or partner, you can always use the Document Viewer to learn about the transaction. Besides that, WebSphere BI Connect did nothing more than move files from the FTP root directory to the file system gateway. Outbound, WebSphere BI Connect acted as a smart FTP client with built-in retries. However, WebSphere BI Connect can provide more value by bridging the FTP server directly to other gateways. Assume for a moment that we want to deliver the incoming XML file as an MQ message to any back-end system.
Such a change is easy to implement and demonstrates well the power of WebSphere BI Connect, even when documents are received over FTP. Chapter 15, “Integration with WebSphere Data Interchange” on page 419, shows how to create a JMS gateway so that EDI messages coming from Company A are routed to a WebSphere Data Interchange server. With a few clicks, you can adjust the current configuration so that FTP received XML documents are sent to WebSphere Data Interchange as well. To achieve this, you use these steps:

1. Select **Account Admin → Participant Connections**.
2. Select **Company F** and **Company E** as the participants and click **Search**.
3. When the two active connections are shown, click **Gateways** for the connection between None/customXML/purchaseOrder and None/customXML/purchaseOrder.
4. Ensure that the target gateway is set to the default gateway for Company E, which is the **FileSystemGateway**. From the drop-down box, select **JMSGateway**, which is the gateway that we define and use in Chapter 15, “Integration with WebSphere Data Interchange” on page 419. Click **Save**.

![Figure 11-36   Updating the target gateway](image-url)
5. Send a new XML file using FTP from Company F to Company E. The XML file should now be available in the queue EDI_IN on the queue manager wdi.queue.manager, as shown in Figure 11-37.

![Message Browser](image)

**Figure 11-37** XML file arrived in EDI_IN queue
As an alternative to MQ Explorer, you can use RFHUTIL to inspect the message in this queue. See Figure 11-38.

![EDI_IN](image)

**Figure 11-38  Browsing the queue EDI_IN**

In the configuration that is used in Chapter 15, “Integration with WebSphere Data Interchange” on page 419, the queue EDI_IN is a triggered queue. This means that WebSphere Data Interchange picks up this message as soon as it is delivered. Thus, to browse the message in the queue EDI_IN, you may need to turn off triggering for this queue.

### 11.4 Implementing FTPs

One obvious disadvantage of using FTP to exchange business critical documents is the low level of security. This is inherent to FTP and has been recognized as a short-coming by standards organizations as well. The Internet Engineering Task Force (IETF) has accepted a standard to extend standard FTP with a number of security extensions.
For more information about this standard, RFC2228, go to:

http://www.ietf.org/rfc/rfc2228.txt?number=2228

In short, this standard allows for a Secure Sockets Layer (SSL)-like extension of FTP. Similarly to HTTP and HTTPs, we have now FTP and FTPs. This means that we can add an SSL hand-shake to the FTP communication between FTP client and FTP server. This SSL hand-shake allows for the authentication of the client, server, or both and encryption of the data so that only the intended receiver can decrypt it.

### 11.4.1 Enabling FTPS for the FTP server of Company F

Enabling FTPs and certificate management highly depends on the FTP server that is being used. In our lab environment, both Company E and Company F use the FTP Server WFTPD Pro V3.2.1 for Windows. This product uses the Microsoft Windows security and encryption features. That is, the FTP server looks for its certificate in the certificate store that is associated with the computer or with the system user. To manipulate these certificate stores, additional Microsoft Windows software components need to be installed and configured.

#### Install .NET run-time and SDK

To use Windows tools for certificate creation and manipulation, you need the .NET Framework and SDK. You need to install the following packages.

- Microsoft .NET Framework 1.1 Redistributable package
- Microsoft .NET Framework SDK Version 1.1
- Microsoft .NET Framework SDK Version 1.1 Service Pack 1
- Update tool makecert.exe (version 5.131.3617.0).

For information about where to obtain the .NET packages and how to install them, see Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57. The only difference is that it is mandatory to install Service Pack 1 for .NET. This is not required in Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57. The easiest way to obtain and install this Service Pack is to use the Windows Update feature. Alternatively, you can visit the Microsoft Web site at:

B-A83353618B38

The updated tool makecert.exe can be downloaded from Microsoft at:

http://download.microsoft.com/download/platformsdk/Update/5.131.3617.0/NT45XP/E
N-US/makecert.exe
Copy the newer tool makecert.exe to the bin directory of the Microsoft .NET SDK, for example C:\Microsoft.NET\SDK\V1.1\bin.

**Generating a CA certificate**

In production environments, you request a certificate from a trusted external Certificate Authority (CA). However, for testing and learning purposes, it is common to use self-signed certificates or to be the CA yourself. Remember that we also used self-signed certificates for WebSphere BI Connect when enabling encryption and digital signatures for AS2.

1. Open a command window, change to the bin folder within the Microsoft .NET SDK installation directory, and execute the following command:

   ```cmd
   makecert -pe -n "CN=WBICFTP_CA" -ss MY -sr CurrentUser -a sha1 -sky signature -r WBICFTP_CA.cer
   ```

   This command creates a certificate and stores in the personal certificate store called MY. It also saves it in the WBICFTP_CA.cer file.

2. Open Windows Explorer. Then locate and double-click the `WBICFTP_CA.cer` file that was generated by the `makecert` command. This enables you to inspect the certificate information, as shown in Figure 11-39.

   ![Certificate Information](image)

   **Figure 11-39 Certificate window**
3. Select the **Details** tab and review the information.

![Certificate dialog box with details of the CA certificate]

**Figure 11-40  Details of the CA certificate**

4. Return to the **General** tab and click **Install Certificate**.
5. The Certificate Import Wizard starts. In the Welcome window (Figure 11-41), click Next.

Figure 11-41  Certificate Import Wizard: Welcome panel
6. In the Certificate Store panel (Figure 11-42), follow these steps:
   a. Select the **Place all certificates in the following store** option.
   b. Click **Browse**.

---

![Certificate Import Wizard: Certificate Store panel](image)

*Figure 11-42  Certificate Import Wizard: Certificate Store panel*
i. In the Select Certificate Store window (Figure 11-43), select the **Show physical stores** option.

ii. Expand the tree structure to locate **Trusted Root Certificate Authorities**.

iii. Select **Local Computer**.

iv. Click **OK**.

![Select Certificate Store](image)

*Figure 11-43  Loading the certificate in a local computer as a trusted CA*
c. Back in the Certificate Store panel, you see the certificate store as shown in Figure 11-44. Click **Next**.

![Certificate Import Wizard: Certificate Store panel](image)

*Figure 11-44  Certificate Import Wizard: Certificate Store panel*
7. In the Completing the Certificate Import Wizard panel (Figure 11-45), click **Finish** to end the wizard.

![Certificate Import Wizard](image)

**Figure 11-45 Certificate Import Wizard: Completion panel**

**Creating the SSL server certificate**

Given that we have now made ourselves a CA, we can issue certificates for other people or purposes, such as FTP servers.

1. Open a command window and type the following command:

   ```
   makecert -pe -n CN=WBICFTP_Server -ss MY -sr LocalMachine -a sha1 -sky exchange -eku 1.3.6.1.5.5.7.3.1 -in WBICFTP_CA -is MY -ir CurrentUser -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12 WBICFTP_Server.cer
   ```

   **Tip:** You may want to write this command in an editor and save it as a command file.

   This command generates a new certificate and stores it in the certificate store of the local computer. The certificate is signed by the CA whose certificate is WBICFTP_CA.

2. Open the configuration of the FTP server.
3. In the Configure Server wbictfp window (Figure 11-46), complete these tasks:
   a. Select the **TLS/SSL** tab.
   b. For **User server certificate from**, select **Machine store**.
   c. In the **Certificate name to use** list, select the newly generated certificate **WBICFTP_Server**.
   d. Select also the encryption options.
   e. Click **OK**.

![Figure 11-46   Associating a server certificate with the FTP server](image)

Notice that this FTP server product does not yet support client certificates. As such, this is not a problem since the client has to authenticate himself using a user ID and password. Since this exchange of the user ID and password is encrypted as well (first encryption option in Figure 11-46), there is no problem.
11.4.2 Uploading certificates in WebSphere BI Connect of Company E

Since we use only a server-side certificate, we must ensure that the certificate of CA is available to WebSphere BI Connect. Since Company F created his own CA, which is the equivalent of self-signed certificates, we need to upload the CA certificate WBICFTP_CA.cer to WebSphere BI Connect. CA certificates or root certificates are uploaded to WebSphere BI Connect in the profile of the Operator.

When this file is received by Company E, the hubadmin user can upload it.

1. Select **Account Admin → Profiles → Community Participant**.
2. Click **Search** and click the magnifying glass icon next to the profile of the Hub Operator.
3. Select **Certificates**.
4. You should now see the list of certificates that are currently used by Company E. Click **Load Certificate**.

![List of certificates in the profile of the Operator](image)

*Figure 11-47  List of certificates in the profile of the Operator*
5. In the Create New Certificate window (Figure 11-48), complete these tasks:
   a. For Certificate Type, select **Root Certificate**.
   b. Type a description for the new certificate.
   c. For Status, select **Enabled**.
   d. For Certificate, click **Browse** to locate the certificate on your hard disk.
   e. Click **Upload**.
6. The WebSphere BI Connect server analyzes the certificate and presents the user with several pieces of information about the certificate. Review it carefully and click **Save**. Figure 11-49 shows the new list of certificates in the profile of the Operator.

![List of certificates in the profile of the Operator](image)

**Figure 11-49**  List of certificates in the profile of the Operator
11.4.3 Updating the FTP gateway

The next change to the WebSphere BI Connect server is to update the existing FTP gateway, which is shown in Figure 11-50.

Figure 11-50 Existing FTPGateway in the profile of Company F
Chapter 11. Integrating FTP servers with WebSphere BI Connect

The gateway FTPGateway, defined in the profile of Company F, uses standard FTP as its transport. Open this gateway and change this transport to FTPS. Click Save.

You do not need to change the URI from ftp to ftps. You would make a change from http to https to use SSL in combination with HTTP. However, this is not the case for FTP. FTP and FTPS are usually linked to the same port, while HTTP and HTTPS are not. See Figure 11-51.

Figure 11-51   Updating the gateway to use FTPS as transport
11.4.4 Validating outbound communication

To validate the outbound communication, you can use the same procedure as outlined in 11.2.7, “Validating communication” on page 298. You can again use the document viewer to inspect the results of the communication.

This time, when the FTPS client of WebSphere BI Connect contacts the FTP server of Company F, it sends this FTP server the `auth` command. The FTP server responds by sending its certificate. This certificate is validated by WebSphere BI Connect. Therefore, it needs the certificate of the CA that has signed the certificate of the FTP server. This is the certificate that we upload as a root certificate in WebSphere BI Connect. The FTPS client of WebSphere BI Connect uses this certificate to encrypt any further communication between the client and the server, including the command `user` and `pass`, which are used to pass user ID and password by the client to the server.

At the end, the upload of the file should not be much different. The file should be available in the home directory of the user company_e on the FTP server of Company F. The success of the operation can again be seen in the Document Viewer.

11.4.5 Using FTPS for inbound communication

Configuring the FTP server of Company E to use FTPS is purely an exercise in configuring the FTP server product. As such, it is not configured in this redbook. If you are interested in performing this task, refer to the guidance in the configuration for FTPS of the FTP server of Company F, which is discussed in 11.4.1, “Enabling FTPS for the FTP server of Company F” on page 320.
11.5 Summary

At the end of this chapter, Company E can now exchange documents with Company F over FTP and possible FTPS. From an operational perspective, we can still use the document viewer to monitor the document exchange with Company F, in the same way that was done for the exchanges with Company A and Company X.

The use of FTP has also little impact on back-end applications that generate the documents for Company F. These applications do need to write the documents in the right directory and the files need to have a specific name. Some portions of the routing logic is offloaded from WebSphere BI Connect to the back-end applications.

When Company F decides to use AS2 as well, we will update the gateway in the profile of Company F to be an HTTP gateway. If you do not want to change the back-end applications and if you want to continue to write documents for Company F in the appropriate directories for Company F, then also replace the FTP Directory receiver with a standard File System receiver, similar to the File System receiver that we use to monitor the \WBIConnect\data\companyf_edi_out directory.
Managing the B2B exchange

This chapter introduces you to some of the tools that are part of WebSphere BI Connect and that can help the administrator to manage the WebSphere BI Connect system. The main tools are:

- Event Viewer
- Document Viewer
- AS2 Viewer

In this chapter, you also learn how to use these tools to investigate a number of problems. You see how these tools provide critical information to find the root cause of a problem.
12.1 Overview of tools to manage the exchange

There are many tools and sources of information within WebSphere BI Connect to manage the system. These tools are available at a number of levels. Low-level sources of information, such as WebSphere Application Server log files, can be useful to investigate low-level problems. Other tools provide a chronological list of events that have occurred, or are happening right now, in the system. Higher-level tools look at documents as they pass through WebSphere BI Connect and events as they are related to a certain document. And there are tools in the system that are specific to certain protocols, notably AS2 and RosettaNet.

All these tools are run-time and real-time tools. They show what happens in the system during a certain time interval, including real-time information. Tools are also available that can perform historical analysis or that can be used prior to going live with a connection. In earlier chapters, we use the Test Connection feature to validate up to some level that the configuration for a certain participant connection is working. You can also use this Test Connection feature as a quick test when problems are reported to the system administrator.

A more extensive test feature exists for RosettaNet interactions. WebSphere BI Connect provides a participant simulator that allows you to validate the setup specifically for RosettaNet.

To perform historical analysis or to perform some type of charge back to business units using the WebSphere BI Connect system, the system administrator can generate a document volume report. These reports can be exported from WebSphere BI Connect as a comma-separated file, which can be imported in a spreadsheet for further analysis.

In addition to wanting to know how many documents have been processed, you may also want to investigate archival options for all completed transactions. A trace of transactions, completed and on-going, is kept in the database and on the file system. For example, if a document was received by WebSphere BI Connect, the system maintains a record in the database about that document, and the actual document (or payload) is stored on the file system. On systems with a reasonable volume of transactions, this can result in a huge amount of data. A procedure to archive this data is documented in the Administrator Guide, which you can download from:

12.2 System log files

Two types of log files exist for each component of the product. The first type of log files are those related to and managed by WebSphere Application Server. For each component directory (receiver, router, and console), there is a folder called was\logs\server1. This folder contains the files SystemErr.txt and SystemOut.txt. Most of the information in these files relates to WebSphere Application Server itself. But when investigating system-related issues, it is worthwhile to check the contents of these files.

In the same folder, you can find a log file managed by WebSphere BI Connect. The file names are wbic_router.log, wbic_receiver.log and wbic_console.log. The contents of this file can be helpful to locate configuration errors or resource problems. By default, the level of detail of these files is limited to errors. But if required, you can increase the output by updating the log4j properties in the properties file for each component. The properties file is located in the was\wbic\config folder within the folder for each component of WebSphere BI Connect.

12.3 Event Viewer

The Event Viewer is probably the first tool that the hub administrator uses to learn how the system is performing and whether any errors have occurred. It is also a viewer that has an automatic refresh rate built-in.

Many filtering options are available in the main window of the Event Viewer. You can filter for several levels of event information, from debug level up to critical. You can also filter on participants, time and date intervals, event location, and so on.
Events are normally shown in a chronological order and are not necessarily linked to a document flow or exchange. The default search arguments are always set to show you all events of the last ten minutes (Figure 12-1).

Figure 12-1  Main interface of the Event Viewer
When an event is found that needs further investigation, the Event Viewer can link to the Document Viewer. Figure 12-2 shows a warning message indicating that a delivery has failed. A message was being sent from Company E to Company A, but it did not succeed.

![Example of a warning event](image-url)
When opening the details of that event, shown in Figure 12-3, you can now open the document that generated this event.

![WebSphere Business Integration Connect Community Console](image)

**Event Details**

- **Event ID**: 1092383261844000C29B10BBB015480C0000000000000167
- **Event Code**: BCG250011
- **Event Name**: First Delivery Attempt Failed
- **Timestamp**: 9/12/04 11:27:41 PM
- **Type**: Warning

**Source IP**: 9.42.170.81

**Location**: System

**Source Participant**: Company E

**Target Participant**: Company A

**Description**: First delivery attempt failed for message 1092383261844000C29B10BBB015480C0000000000000164 due to ", on gateway 'HTTPGateway@[unspecified]'" (id=2)

**Document Details**

- **Document ID**: 000000001

![Document Details Table](image)

**Participants** | **Time Stamps** | **Document Flow** | **Doc Time Stamp**
---|---|---|---
Source: Company E | In: 8/12/04 11:27:39 PM | EDD-X12: ALL ALL (ALL) | 
Target: Company A | Out: 8/12/04 11:32:43 PM | EDD-X12: ALL ALL (ALL) | 961007-2013

*Figure 12-3  Details of the event and the associated document*
In the Document Details section, in Figure 12-3, select the ** icon. This opens the window shown in Figure 12-4.

![WebSphere Business Integration Connect Community Console](image)

**Document Details**

| Reference ID: | 10923526587340000029556600000000000000161 |
| Document ID: | 00000000 |

<table>
<thead>
<tr>
<th>Doc Time Stamp</th>
<th>Gateway Type</th>
<th>Connection Document Flow</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>961007-2013</td>
<td>Production</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>In Time Stamp</th>
<th>Source Business ID</th>
<th>Source Document Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Company E" /></td>
<td>8/12/04 11:27:39 PM</td>
<td>compayne</td>
<td>EDI-X12 ALL: ALL ALL ALL</td>
</tr>
<tr>
<td><img src="image" alt="Company A" /></td>
<td>8/12/04 11:33:43 PM</td>
<td>compayne</td>
<td>EDI-X12 ALL: ALL ALL ALL</td>
</tr>
</tbody>
</table>

**Document Events**

<table>
<thead>
<tr>
<th>Total Event Count:</th>
<th>6</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Timestamp</th>
<th>Type</th>
<th>Event Code</th>
<th>Location</th>
<th>Source IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Delivery Attempt Failed</td>
<td>8/12/04 11:27:41 PM</td>
<td>Warning</td>
<td>BCG350011</td>
<td>System</td>
<td>9.42.170.81</td>
</tr>
</tbody>
</table>

* Figure 12-4 Details of a document with associated events

Where Figure 12-3 shows the event and the associated document, Figure 12-4 shows the document and its associated event count. Now it becomes apparent that there are other events linked to this same document. By changing the filter level to Information or Debug, all events should now be shown.
Figure 12-5 shows a filtered view of all events that relate to this document. This time, you can see that the document was delivered, after a first delivery attempt failed.
12.4 Document Viewer

The Document Viewer is used to look at documents as they pass through the system. You can use it to look at documents in different states of transition, documents that include specific partners, and many other filtering options, as shown in Figure 12-6.

Figure 12-6   Search options for the Document Viewer
Figure 12-7 shows the resulting list of documents for a particular time interval. The lower document was sent from Company E to Company A. During its processing within WebSphere BI Connect, the document grew from around 1.5 KB to more than 2 KB. This is the result of the packaging required for sending the document.

Above that document, you see another document, which at first glance is unrelated to the lower document. This time, a document was received from Company A by Company E. There is no associated unpackaged document. If you click the icon, you see again the details of the document and any events associated with that document.
When clicking the icon, you see the raw document. You notice that the top document is actually a message disposition notification (MDN). After you inspect the details of the lower (older) document in Figure 12-7 on page 346, you learn that this MDN is linked to the AS2 document at the bottom of the list in that figure. Thus, documents listed in the Document Viewer are not necessarily grouped together. That is the case for the third viewer: the AS1/AS2 Viewer.

### 12.5 AS1/AS2 Viewer

Earlier in this redbook, we used the AS1/AS2 Viewer when we described how to use it to validate the configuration and how to learn about AS2 transactions. See 7.7, “Validating communication” on page 178.

The search options for the AS1/AS2 Viewer are more or less the same as the options for the Document Viewer. Any result is now specific to AS2 packaging. No other types of communication are shown in the AS1/AS2 Viewer. A major differentiator is that now related documents are grouped together. Even more, the initial list of results in the AS1/AS2 Viewer shows one item per AS1/AS2 transaction which includes the original EDI document and the MDN.
As shown in Figure 12-8, for each item in that list, there are the source and target partners, the message ID, and the MDN status. For all documents shown in Figure 12-8, WebSphere BI Connect received a corresponding MDN.

![WebSphere Business Integration Connect Community Console](image)

Figure 12-8  List of AS2 transactions

Click the icon next to a transaction and see all three documents that belong to this transaction. You can inspect the electronic data interchange (EDI) document that was picked up by WebSphere BI Connect in the edi_out directory. You can see the AS2 packaged document that was sent to the partner. This packaged document has the AS2 and HTTP header information.

Click the icon next to the outbound document to see details about that document and packaging. In this window, you see information about the AS2 packaging and AS2 status, the EDI document, and the events that relate to this
When you change the event filter from Warning to Information, you see all possible details, as shown in Figure 12.9.

### Package Details

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Source Participant</th>
<th>Target Participant</th>
<th>Source Time Stamp</th>
<th>Gateway Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1109948651600C2961088501484000860008016525</td>
<td>Company E</td>
<td>Company A</td>
<td>7/16/04 1:17:31 AM</td>
<td>Production</td>
</tr>
</tbody>
</table>

### MDN URI

http://w3ehub:57000/begreceiver/companya/edi_in

### MDN Disposition Text

automatic-action/MDN-event-automatically-processed

### Document Details

<table>
<thead>
<tr>
<th>Doc Time Stamp</th>
<th>Gateway Type</th>
<th>Connection Document Flow</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/16/04 1:17:31 AM</td>
<td>Production</td>
<td>All All All All All All</td>
<td>✔️</td>
</tr>
</tbody>
</table>

### Document Events

- **Total Event Count: 5**

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Time Stamp</th>
<th>Type</th>
<th>Event Code</th>
<th>Location</th>
<th>Source IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Parse Successful</td>
<td>7/16/04 1:17:31 AM</td>
<td>Info</td>
<td>BC021060</td>
<td>Unknown</td>
<td>9.42.178.80</td>
</tr>
</tbody>
</table>

### Event Details

- **Passed destination parse**

- **Destination Parse Successful**
  - 7/16/04 1:17:31 AM
  - Info
  - BC021062
  - Unknown
  - 9.42.178.80

- **Document Sequence Validation Successful**
  - 7/16/04 1:17:31 AM
  - Info
  - BC020001
  - Unknown
  - 9.42.178.80

- **Document Sent to Outbound Processor**
  - 7/16/04 1:17:31 AM
  - Info
  - BC020005
  - Unknown
  - 9.42.178.80

- **Document Delivered**
  - 7/16/04 1:17:32 AM
  - Info
  - BC025004
  - Unknown
  - 9.42.178.80
Figure 12-8 on page 348 shows a list of completed AS2 transmissions. However, the AS2 Viewer can also help to determine which transactions are incomplete, because WebSphere BI Connect has not yet received an MDN for a given document. Figure 12-10 shows an example of such a document. The clock icon in the MDN Status column indicates that the MDN has not yet arrived for this document.

![WebSphere Business Integration Connect Community Console](image)

**Figure 12-10**  AS2 transmission in progress

### 12.6 Gateway queue

The next viewer utility allows you to inspect the delays or the queuing at a gateway. On a lightly-loaded system, you may not be able to find any queuing at a gateway, since the message may be processed before you reach this viewer.

The gateway queue viewer can be configured to refresh automatically. You can configure it also to only show documents with a delay that is longer than a certain number of minutes. This can be useful, for example, to see if service level agreements are being met.
Figure 12-11 shows that one single document is queued at the gateway called HTTPGateway.
When you click the number below the Queued label, you see additional configuration details about the gateway and information about the queued documents, as shown in Figure 12-12.

```
<table>
<thead>
<tr>
<th>Queued Documents Search</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Participant:** Company A  

**Gateway Info**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>State</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPGateway</td>
<td>null</td>
<td>Online</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transport</th>
<th>Target URL</th>
<th>Retry Count</th>
<th>Retry Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/1.1</td>
<td><a href="http://9.42.171.04:57000/companyA/odi_in">http://9.42.171.04:57000/companyA/odi_in</a></td>
<td>3</td>
<td>300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threads</th>
<th>Validate Client IP</th>
<th>Validate Client SSL</th>
<th>Validate Auto Config</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Queued Documents**

<table>
<thead>
<tr>
<th>Reference ID</th>
<th>Participant (Gateway)</th>
<th>DocID</th>
<th>Queued TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10899216540290000C23E10BB6D94840000000000006748</td>
<td>Company A (HTTPGateway)</td>
<td>000000001</td>
<td>7/15/04 8:00:55 PM</td>
</tr>
</tbody>
</table>

```

*Figure 12-12  Details about the gateway and queued documents on that gateway*
When you select the Reference ID in Figure 12-12, you see additional information about the ongoing processing and transfer. This window, shown in Figure 12-13, give you the option to stop the process. If you suspect that something is wrong with a given transmission and that, as a result of such a problem, other documents are getting queued, you can click the Stop Process link in Figure 12-13 to stop the processing of this document.

Figure 12-13  Details about the gateway and a document in the gateway queue
12.7 Using the tools to solve problems

The previous sections demonstrate the use of a number of tools that a system administrator can use to find problems, look up documents in transit, or review the status of AS2 exchanges. This section describes the use of these tools to solve a few problems.

12.7.1 MDN HTTP URL not defined

An outbound AS2 document needs a URL for the MDN response. If an AS2 document should not have this, then the receiving partner does not know how to send an automated response. To avoid this situation and to make sure that the packaged AS2 document adheres to the AS2 standard, WebSphere BI Connect verifies this MDN URL and other attributes of an AS2 connection.

Incorrect or incomplete settings such as that are considered critical errors. As such, the packaging within WebSphere BI Connect does not go far enough to have a document classified as an AS2 document. If you try to send a document over AS2 and nothing appears in the AS2 Viewer, try the Event Viewer, which is the first place where you may find information about such a problem. Figure 12-14 shows a critical event that refers to an AS Packager Failure.

![Figure 12-14  Critical event related to AS packaging](image)
When opening the details of this event, you see a window similar to Figure 12-15. The description field clearly explains what is wrong. The document details also provide information about the document flow (EDI-X12) and the participants. With this information, the hudadmin user or the administrator of Company E has sufficient information to look up the participant connection and to correct the value for the attribute AS MDN HTTP URL. Refer to 7.5, “Connecting Company E to Company A” on page 163, and to Figure 7-43 on page 169.
12.7.2 Problems with encryption

When implementing encryption in a WebSphere BI Connect environment, you may have to correct a few errors before you set it up correctly. Since encryption is performed at the AS2 packaging layer, you may find that the AS2 Viewer has no information about the document that you try to send. When encryption fails, it is considered a critical packaging error that appears only in the Event Viewer, as shown in Figure 12-16.

![Event Viewer](image)

Figure 12-16 Critical event related to AS2 packaging
When opening up the details of this event, you see the description of the event, as shown in Figure 12-17. The error text says that the certificate is null, which is caused by not having the right certificate in the right place. When Company E needs to send an encrypted document to Company A, the server looks for the public key of Company A in the profile of Company A. If you accidentally upload the public key in another profile, such as the hubadmin profile, you may see a similar error.

![Image of Event Viewer](image)

**Figure 12-17** Details of the critical event

Besides the Event Viewer, this type of error is also reported in the Document Viewer. When searching for failed documents, this error is shown as well.
12.7.3 Problems with digital signatures

When implementing digital signatures, you may again experience a situation where documents are not processed the way you expect. Figure 12-18 shows a warning message that at first seems not so critical. Company E receives an MDN from Company A that is not signed, while signed MDNs were requested.

Figure 12-18  Warning event about unsigned MDN
When you open the details of this event, shown in Figure 12-19, you have access to the document that is linked to this event. It shows that the MDN was sent by Company A. Click the "icon for that document to look at the MDN as it has been received by Company E.

Figure 12-19   Details of the event and the linked document
The actual MDN, shown in Figure 12-20, contains a more interesting problem. The disposition message says that the authentication has failed. This can explain why Company A did not bother signing the MDN. Company A was unable to authenticate Company E as the sender of the document. What is this authentication to which the MDN is referring?

Figure 12-20  Details of the MDN that was unsigned
Now look at this same problem from an AS2 perspective. When using the AS2 Viewer, it lists this transaction with a status of MDN received, as shown in Figure 12-21. At first, there does not seem to be anything wrong with it.

Figure 12-21  Overview of the AS2 exchange that caused the warning message
When you open the details for this specific AS2 exchange, you have the sent document and the received MDN in one browser window. This time, you see the disposition text in the MDN, which we saw earlier. The fact that the MDN has been received is not necessarily an indication of a successful transmission. There are positive MDNs and negative MDNs. Thus, the contents of the MDN must be inspected as well (Figure 12-22).

![WebSphere Business Integration Connect Community Console](image)

**Figure 12-22** Details of the AS2 exchange as reported by the sender Company E

At this moment, we still do not know why authentication has failed. Also, if you use the Document Viewer of Company E (the sender), you do not see this document listed as a failed document. Since an MDN was received, WebSphere BI Connect assumes that the processing was successful.
Let's look closer at the server of Company A and see how this problem may have been handled by the receiver of the document that failed authentication. Using the Event Viewer at Company A, an error event is being logged. When opening the details of that event, shown in Figure 12-22, the kind of authentication that has failed becomes clear. The signature that was included in the original AS2 document could not be verified (see Figure 12-23).

![Event Viewer Screen Shot]

**Figure 12-23** Details of the event as reported by the receiver Company A

When we use the AS2 Viewer on the server of Company A, we see that the AS2 transaction is listed as a transaction in error. The unencrypted EDI document is not available, only the encrypted document is as it was received at the HTTP protocol level. The MDN that was sent back to Company E is listed as well (Figure 12-24).
If we open the Document Viewer on Company A, then we find that this specific document is categorized as a rejected document. So, what went wrong in this case? What is required to validate a digital signature? Two things are needed:

- The public key of the sender, Company E, needs to be available in their profile on the server of Company A. It must also be marked as used for digital signatures.
- The public key of the sender, Company E, needs to be uploaded to the profile of the user hubadmin and marked as used for CA Root.

Because this last step was not performed, the error condition discussed here was the result. This last step is required because we used self-signed certificates. For certificates issued by a commercial Certificate Authority, we would have uploaded the public key of the CA itself.
Integration with WebSphere Data Interchange
Introduction to EDI technology and WebSphere Data Interchange

This chapter provides a brief overview of the electronic data interchange (EDI) concept and technology. It also describes WebSphere Data Interchange and how the product fits into a typical EDI solution.
13.1 EDI terms and concepts

Electronic data interchange is a concept that has been in commercial use for more than 30 years. It is widely accepted by companies all over the world as the way to electronically exchange business data.

Over the years, we have seen a variety of interpretations of the term EDI. A common and basic definition of EDI is:

“The transfer of business data between computer applications using a mutually agreed standard to describe the data contained in the message”

Typically this means that business data is extracted from a company’s internal application in an application-specific data format. This data extraction can be implemented in several ways. It can be a daily batch job reading information in a database and generating a file in an application-specific format. This user format data file is then translated into a standardized format such as EDI or XML and transmitted over a network to a trading partner.

An alternative technique is to generate WebSphere MQ messages from within the application. That WebSphere MQ message is again in an application-specific format.

For both techniques, the message ends in some way at a translator component, where the application-specific structure of the data is mapped to an EDI standard format, as shown in Figure 13-1.

![Figure 13-1 The role of the translator](image)
The receiving trading partner then retranslates the received EDI or XML message back into an application-readable format that fits into their processes, as shown in Figure 13-2.

EDI-based information exchange is usually a two-way process. Thus, the translator component is also used to translate incoming EDI messages into an application-specific format.

![WebSphere Data Interchange Typical Data Flow](image)

*Figure 13-2  Basic EDI message flow showing WebSphere Data Interchange*

From a company perspective, the EDI concept means business integration and process automation. Business documents, such as purchase orders, invoices, shipping notices, and price catalogs are exchanged between companies over a network in a structured and computer processable format.

Figure 13-3 shows a typical flow of actions and data between a buyer and a seller. Usually, a buyer asks for a quote. Then when a quote is received, a purchase order request may be sent by the buyer to the supplier. This information exchange is typically handled by a purchasing application at the buyer side and handled by a sales management application at the seller side.

When the goods are ready to be delivered, a shipping notice is sent by the seller to the buyer. This time, the information exchange is likely between different components of the IT infrastructure at both the seller and the buyer side.

Thus, the use of EDI between two companies implies integration between the applications at each end. The purchase order generated by the purchase application needs to be known by the application used in the warehouse or by the accounting application.
Since the early days of EDI, many new initiatives and techniques have been adopted by the market. Words that hardly existed at that time, such as the Web, XML, B2B, and business process management (BPM), are a natural part of today’s realities. The obvious question is, why is EDI still so important?

- EDI is an important part of companies’ B2B strategies.
- Of Fortune 500 companies, 95% use EDI.
- A total of 80% of business transactions are conducted via EDI value added networks (VAN) today.
- EDI continues to deliver a significant return on investment.
- EDI continues to evolve in response to new enterprise and industry requirements, and competitive pressures (for example, HIPAA, AS1, AS2).

### 13.2 Benefits of EDI

The market is driving every business to act smarter and quicker and to be more visible. Much of this can be achieved using EDI. Even better, EDI can give companies a better knowledge of their markets, because it opens possibilities to collect and analyze information from the EDI transactions they are generating.
Among the most visible benefits of adopting EDI are:

- Reduction of data entry errors
- Reduced cycle time
- Minimization of paper use
- Improved relationships with your business partners
- Information in electronic form is more easily shared over the organization
- Improved inventory management

### 13.3 EDI components

The term EDI is a concept. It does not define any technique, nor point to any specified product or service. An EDI transmission can be divided into two logical parts: the message itself and the communication.

#### 13.3.1 Message standards

Since the idea of EDI is to have a standardized message, several different standards have been developed and established over the years. The most commonly used message standards are:

- **ANSI ASC X12**: U.S. standard
- **EDIFACT**: Standard recommended by the United Nations, used mainly in Europe
- **UNTDI**: U.K. retail standard
- **ODETTE**: European automotive industry
- **Others such as HIPAA, VICS, VDA, and UCS**

The standardized messages are built by components such as elements, segments, and transactions/messages. Between every object there is a separator.

The *elements* are the individually defined fields such as amount, name or quantity. Two or more elements can be grouped together, forming a *composite element*.

A *segment* is a set of elements or composite elements built to a logical entity such as name and address or pricing information.

An *envelope* contains overall information about the transaction or message, such as sender and receiver, type, and control values.

A set of segments put together in a specified order all wrapped in an envelope make a *transaction* or *message*, such as an invoice or a purchase order. The envelope contains information about sender and receiver, transaction/message type, and so on.
Figure 13-4 shows this structure of an EDI message in a graphical way.

Example 13-1 shows a sample X12 transaction. The first three lines are part of the envelope. The line starting with ST*810 is the start of the actual message. This time it is an 810 message, which is used to send invoices.

**Example 13-1 X12 transaction (invoice)**

```
ISA*00*  *00*  *ZZ*CELRGC02  *ZZ*IBMIRLPROD  *021018
*0229*U*0401*00089990*0*P**-
GS*IN*CELTALY*IBMIRELAND*20021018*0229*8899*X*004010-
ST*810*88990001-
BIG*20021017*0002146553**P350342***DR-
CUR*SE*USD-
REF*D2*0080118614-
N1*SE*Celestica Italia S.r.l.*92*103015-
N2*Celestica Italia S.r.l.-
N3*Via Lecco 61-N4* Vimercate - MI - IT**20059-
REF*GT*IT03029690967-
N1*BY*IBM INTERNATIONAL HOLDINGS-
N2*IBM INTERNATIONAL HOLDINGS-
N3*MULHUDDARTH-
N4*DUBLIN*DB*15-
REF*GT*I56602632V-
IT1*000001*4*EA*1767.87*PE*BP*00004N3524*VP*4N3524-
TXI*VA*0*0-
PID****BK_C_F_CARDINAL-
REF*ZZ*7071.48-TDS*707148*707148-
TXI*VA*0-
CTT*1-
```
Both the X12 and the EDIFACT transactions in Example 13-1 and Example 13-2 are presented with one segment per row to be easier to view. Normally a new segment follows directly after the previous segment to save space.

**Example 13-2  EDIFACT message (Purchase Order)**

<table>
<thead>
<tr>
<th>Segment</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNB+UNOA:2+3568579005454:14+3015437860102:14+021003:0053+02018852760++ORDERS'</td>
<td></td>
</tr>
<tr>
<td>UNH+1+ORDERS:D:93A:UN:EAN007'</td>
<td></td>
</tr>
<tr>
<td>BGM+220::9+001779'</td>
<td></td>
</tr>
<tr>
<td>DTM+137:20021002:102'</td>
<td></td>
</tr>
<tr>
<td>DTM+2:20021005:102'</td>
<td></td>
</tr>
<tr>
<td>DTM+63:20021005:102'</td>
<td></td>
</tr>
<tr>
<td>NAD+BY+3568579005454::9'</td>
<td></td>
</tr>
<tr>
<td>NAD+SU+3015166100102::9'</td>
<td></td>
</tr>
<tr>
<td>NAD+DP+3568579005454::9'</td>
<td></td>
</tr>
<tr>
<td>CUX+2:EUR:9+3:EUR:4'</td>
<td></td>
</tr>
<tr>
<td>LIN+1++3560998032054:EN::9'</td>
<td></td>
</tr>
<tr>
<td>QTY+21:2'QTY+59:1'</td>
<td></td>
</tr>
<tr>
<td>PRI+AAA:798.33::NTP'</td>
<td></td>
</tr>
<tr>
<td>LIN+2++3560998032054:EN::9'</td>
<td></td>
</tr>
<tr>
<td>QTY+21.5'QTY+59:1'</td>
<td></td>
</tr>
<tr>
<td>PRI+AAA:34.6::NTP'</td>
<td></td>
</tr>
<tr>
<td>UNS+S'</td>
<td></td>
</tr>
<tr>
<td>UNT+17+1'</td>
<td></td>
</tr>
<tr>
<td>UNZ+1+02018852760'</td>
<td></td>
</tr>
</tbody>
</table>

From a message organization point of view, both look similar. Every segment starts with a three-letter word indicating the type of segment that follows. Each element within the segment is separated from the next one by a separator. Finally, the message structure uses a segment terminator. There are more rules than this. Some elements are optional, while other elements are conditional. Element A and B are labeled conditional when, for example, the appearance of element A implies the appearance of element B.

### 13.3.2 Communication

Transportation of the EDI file over a network can be done in many ways. You can use any network and any protocol as long as it fits the needs. This section looks at three types of communication:

- VAN communication
- Internet (AS1, AS2, FTP...)
- WebSphere MQ
This section focuses more on the communication aspect between two trading partners. There is also a communication aspect within the IT setup of a trading partner. The data has to be sent from the internal applications to the EDI translator software. After translation, the data must be handed over to some communication software.

**VAN communication**

Using a VAN for the transmission of files is traditionally seen as the most secure way of communication (Figure 13-5). Apart from doing pure communication, a VAN also provides value adds such as:

- Built-in security features that help protect against unauthorized access to customer data
- Restart and recovery facilities that help to reduce or eliminate the impact of communications interruptions
- Archive capability for the online retention of data copies
- 24x7 availability
- Notification of message arrivals that meet predefined criteria, such as a message from a specific trading partner

![Figure 13-5 EDI VAN network](image)

The VAN Gateway software drops off and picks up EDI documents via their mailbox. The VAN provides store-and-forward mailbox services. The physical communication system between the VAN Gateway and the VAN network can vary from dial-up to FTP, or some proprietary communication technology.
IBM Information Exchange (IIN) is an example of such a value added network.

**EDI over the Internet**
The initiative to move toward securely transmitted EDI messages over the Internet is known as EDI INT. Currently there are two main EDI INT initiatives, known as applicability statements AS1 and AS2. They describe how current Internet standards can be used to achieve VAN functionality.

- AS1 uses MIME and SMTP.
- AS2 uses MIME and HTTP for process-to-process real-time EDI.

The Internet solutions are often considered much cheaper than traditional VANs. However, Internet solutions often leave it to the user to add functionality to achieve adequate security, reliability, and other features that are included in a VAN.

IBM Business Exchange Services - Internet transfer is an example of Internet communication.

**Message queuing**
Message queueing (MQ) connects commercial systems in today’s business. It provides assured, once-only delivery of data in any format.

IBM WebSphere MQ is an example of this.

### 13.4 The evolution of EDI

In today’s economy, market dynamics have converged on a business model that provides for the integration of different trading partners in a value chain. Depending heavily on Internet technologies, this model can enable highly coordinated trading communities, each with the ability to operate as a virtual enterprise.

In the virtual marketplace, business relationships are formed electronically. Buyers and sellers come together without the benefit of paper contracts, fee schedules, or sales people to close the deal. This Web economy requires an agile enterprise, one that can work more directly with suppliers and customers and respond more rapidly and intelligently to change. The need for flexibility and lower costs, such as VAN charges, are driving the evolution of EDI.

Organizations are recognizing the value of many years of investments in EDI. Rather than replace the present solution, they plan to extend and evolve the EDI transactions. This existing EDI solution is considered as a part of a multi-modal B2B gateway or hub alongside XML, Web solutions, and portals. By integrating
B2B and EDI technologies, event-driven or process-driven integration models can be supported using the existing EDI solution (Figure 13-6).

The Internet is widely perceived as being much less expensive than a VAN, but this is not necessarily the case. VANs generally provide valuable services, such as Trading Partner Agreement (TPA) management, service-level administration, security, and store-and-forward capability. The Internet requires you to manage these elements yourself, which means the total costs are not always lower than a VAN.

EDI users cannot realize the full value of the Internet in e-commerce applications until the entire underlying business process is optimized. BPM is the automation, optimization, and management of end-to-end business process flows. And, in this case, it is accomplished by integrating front-end Web applications to back-end legacy applications and to existing EDI trading partners.

Earlier phases of EDI achieved efficiency by automating manual processes. Now, however, the focus is on business process integration and optimizing business operations. EDI steps are tied into the full value-chain processes with the ability to share information throughout.
The result of these trends is that traditional EDI customers are facing increasing challenges to remain competitive. To grow or even preserve your business, you need to integrate your existing EDI applications with core business processes by distributing transactions or information to and from various back-end applications. Situations in which this kind of integration can help are:

- Where you have typically spent tens or hundreds of thousands of dollars on your current EDI solutions and you want to leverage that investment
- When internal departments lack timely information about EDI transactions and make costly mistakes or give poor service
- Where competitiveness suffers from an inability to track or manage the distribution of EDI messages within the business
- When manual processing of EDI messages is slow and error prone and consumes valuable resources

13.4.1 Elements of an EDI solution

In addition to obvious components of an EDI solution, such as application programs and systems, VANs, and trading partners, a complete and flexible solution should include the following important elements.

**Translators**

A universal problem in the integration of applications is the conversion of shared data from one format to another. Common data fields, such as names, addresses, and numbers, often have different formats across disparate systems. The traditional approach to EDI implementation is to place the function that converts application data to the EDI standard directly into the business application. This approach is less effective because a separate program is required for each transaction as well as for each trading partner. In addition, it is difficult to keep up with new versions of standards because programs must be modified every time a trading partner adopts a newer standard or version of the standard.

This approach has changed with the introduction of third-party translation software, also known as *mappers*. The translator is responsible for mapping application data to the specific EDI format and vice versa. This translation software is implemented in either a centralized engine or in an adapter. It must handle primary EDI standards as well as different and evolving versions of each standard.
**Batch enveloper/de-enveloper**

Typically, because VAN charging is based on each sent transaction, enterprises have been driven to find ways to reduce the number of transactions and to compress more information into each. Consequently, EDI messages are sent in large batches, which can then be grouped from, or split out to, several divisions or areas of an enterprise.

Enveloping batch messages involves placing the EDI standard header and trailer around transactions in preparation for sending. When the envelope is complete, the package can then be sent to a trading partner through a VAN. Similarly, batch transactions must be deenveloped when they are received from the VAN.

**Message router**

When the EDI message is de-enveloped, it can be divided into function groups. Each function group may relate to a different division or area of the business. A mechanism is needed to sort messages destined for different groups and deliver them to the appropriate target applications. This means there is a requirement to fan in and fan out messages. Message transformation may also be required to convert the message into the correct format for the end applications.

**Trading Partner Agreements**

A TPA is an agreement related to the exchange of information in electronic transactions. The term includes a particular agreed-upon standard for business documents as well as communications and business protocols, the service-level agreement, and more. TPAs can also be extended to include business events. For example, if an event occurs in one organization that may affect processes in a second organization, the TPA can specify that the second organization be alerted to the event.

### 13.4.2 The IBM EDI solution

The key to the IBM EDI solution, shown in Figure 13-7, is IBM WebSphere Data Interchange for Multiplatforms. It is the core of the solution and handles key EDI solution elements, such as translation and enveloping/de-enveloping.

Working closely in partnership with an integration broker, WebSphere Data Interchange can:

- Automate the distribution of EDI messages to and from all departments and trading partners.
- Transform EDI messages, on the fly, to the proper format for each existing application.
- Automatically redirect messages based on message content and system state.
- Track the flow of messages through systems for offline analysis and data mining.
- Reconfigure the system to respond to changing circumstances, by adding or deleting applications.

![Image of IBM EDI solution](image)

**Figure 13-7  The IBM EDI solution**

### 13.4.3 Features of WebSphere Data Interchange

WebSphere Data Interchange for Multiplatforms V3.2 provides advanced translation, validation, and batched information exchange capabilities for EDI standards and for XML. It electronically translates EDI format data, such as invoices, purchase orders, and billing forms, for exchange with trading partners. It supports industry implementations of the ANSI X12, EDIFACT, VICS, UCS and Rail standards.

Translation can take place between any combination of EDI, XML, or structured Application Data Format, which is a feature that is sometimes called *any-to-any transformation*. WebSphere Data Interchange V3.2 provides advanced data validation and standards compliance functions that allow the functional acknowledgments, defined by some standards, to be generated in response to inconsistencies in the data content. WebSphere Data Interchange V3.2 can be configured to both construct and de-construct envelopes of EDI format data that contain batches of related EDI items such as invoices or purchase orders.
WebSphere Data Interchange V3.2 provides a dedicated graphical user interface (GUI) mapping tool, the WebSphere Data Interchange Client, which is optimized to build EDI, XML, and Application Data Format transformations. The WebSphere Data Interchange Client allows direct import of EDI standards definitions, Application Data Format structures and industry standards or user-defined XML document type definitions (DTDs) or schemas for mapping and translation.

The WebSphere Data Interchange Client provides configuration and administration capabilities. Network profiles and trading partner profiles can all be managed via the client interface of WebSphere Data Interchange.

Besides mapping and configuration, the client interface can also be used for auditing and run-time support tasks. The client offers an interface to create reports on transactions and messages. Event logs and activity logs can be created and used as a way to analyze system behavior.

WebSphere Data Interchange V3.2 is available on the Windows 2000, AIX, and z/OS® platforms. The server component is available in two shapes. The first way to use the translation engine is to start in a batch-type of mode driven by a command file. The command file typically contains actions that the translation engine has to perform on several files containing EDI messages. Typical actions include batching, enveloping, and deenveloping, sending and receiving files. Usually the startup of the translation engine is controlled by an automation tool.

Another way to launch the translation engine is using the WDIAdapter program. You can configure WebSphere MQ so that the WDIAdapter program is launched when a message arrives at a queue. The adapter program then reads this message and performs the translations that are required, as configured in WebSphere Data Interchange. The translated message can be written to another WebSphere MQ queue.

The reading and writing of WebSphere MQ messages can be performed using:

- Standard MQ messages with only a message descriptor
- MQ messages with an MQRFH2 header
  The header can contain an mcd folder to indicate message set, type, and format. It can also contain additional information in the USR folder to indicate receiver and sender information.
- MQ messages destined for Java Message Service (JMS) API clients

WebSphere Data Interchange V3.2 supports integration with WebSphere MQ, enabling interoperation with a wide range of enterprise applications, business process engines such as the IBM InterChange Server, information brokers such as WebSphere BI Message Broker, and ERP systems such as SAP R3.
WebSphere Data Interchange V3.2 provides for communication with trading partners via both VANs or Internet B2B gateways by provision of an easy-to-use configurable interface that enables connection to leading VAN and Internet gateways. The WebSphere BI Connect offerings that provide AS1 and AS2 support and IBM On Demand Business, hosting Expedite VAN gateway, are two examples of supported gateways from IBM.

In the context of a typical enterprise integration architecture, WebSphere Data Interchange fulfills the role of an EDI broker that performs the specialist EDI validation, transformation, and exchange functions, and propagates the resulting transformed information either internally or externally. Internal propagation of transformed information may be via a message broker, a process broker, direct to the business applications, or any combination of those depending on the needs of an enterprise. External propagation of transformed information or receipt of information may be via a specialized dedicated VAN gateway or an Internet B2B gateway, or directly to a trading partner. Or it may be via any combination of those interfaces depending on the nature of the trading relationship between the enterprise and its trading partner.

### 13.5 Usage patterns for WebSphere Data Interchange

This section discusses scenarios where WebSphere Data Interchange is used in conjunction with other components of a typical enterprise IT infrastructure.

#### 13.5.1 A point-to-point solution

A first implementation of WebSphere Data Interchange in an enterprise environment consists of a direct link between an enterprise resource planning (ERP) system or other internal application that manages your business, and an EDI broker, such as WebSphere Data Interchange. The communication between those two components can be MQ based or file based. The organization of the data that is being passed between the ERP and the EDI broker can be XML documents or data that can be modeled as C structures or COBOL copybooks.

When the information is translated into the appropriate EDI standard, it is given to a communication product that interacts with the network or VAN.

**Note:** WebSphere Data Interchange is separate from the actual communications infrastructure and can work with various software products. It also provides configuration support for several software products, such as IBM Information Exchange. The interaction between the EDI broker and the communication software can be file based or MQ based.
While this scenario is labeled as point-to-point (Figure 13-8), it still works fine for communication with a number of partners. The point-to-point label refers more to the direct link between a single internal application and the EDI broker instance.

For inbound communication, the EDI information is received by the communication software and handed over the EDI broker via files or queues. The EDI broker then translates the information into the appropriate format for the back-end system.

![Diagram of EDI solution](image)

**Figure 13-8  Point-to-point EDI solution**

### 13.5.2 An integration broker solution

Typically, there is more than one single application system. Referring to Figure 13-3 on page 370, a typical business cycle involves a number of application systems, such as order management, scheduling systems, and so on. You can apply the point-to-point solution in this environment, too. However, as discussed earlier, application integration between the different components of an IT infrastructure is almost a prerequisite for a successful EDI implementation. Therefore, the integration of the EDI broker into the existing EAI infrastructure is an important advantage.

Figure 13-9 shows a schematic view of the integration between a message broker and an EDI broker. The message broker is responsible for distributing information between the different applications. When the broker receives information from one system, it can pass it over to other systems, including the EDI broker.

When, for example, the EDI broker receives a purchase order document, it can translate it into an XML document and hand it over to the message broker. The message broker in turn fans out to the different internal systems, based on message content and system state. The contents of each message generated by the message broker can be set with information out of the incoming EDI message, but can also be enriched by the broker.
13.5.3 A B2B gateway solution

While the use of EDI technology is widespread, technology changes and evolution have resulted in the use of many types of B2B communication infrastructures. Besides the traditional VAN-based EDI communication, Internet-based techniques have become available, too. AS1 and AS2 were mentioned earlier. They are still tied more or less to traditional EDI communications. More recently, Web services-based technologies also became available for use in the B2B area. While this technology is still maturing, it is clear that a flexible B2B solution should handle multiple communication techniques.

A B2B gateway solution, such as WebSphere BI Connect, offers an answer to these challenges. As shown in Figure 13-10, an EDI broker works next to an AS2 solution and a Web services solution. This offers trading partners a wide range of technology options for interacting, and at the same time, there is a single point of control and management for all technologies.

**Note:** The B2B gateway solution can be integrated with the integration broker solution.
13.6 Internet references

You can find more information about the different EDI products from IBM, the WebSphere platform, and the EDI standards on the following Web sites:

- WebSphere Data Interchange
  http://www.ibm.com/websphere/datainterchange

- IBM Information Exchange
  http://ieas.services.ibm.com/ie/index.shtml

- Expedite family
  http://edi.services.ibm.com/expedite

- Business Exchange Internet transfer

- WebSphere software platform

- WebSphere Business Integration Family
  http://www.ibm.com/websphere/integrationinfo/

- EDI standards organizations
  http://www.disa.org
  http://www.unece.org/trade/undid/welcome.htm
WebSphere Data Interchange infrastructure and implementation

This chapter examines the structure of WebSphere Data Interchange and how it uses other software layers to perform its translation job. Using this knowledge, you can easily follow the steps to install and set a WebSphere Data Interchange environment for development and run-time purposes.

This chapter also explains several key concepts within WebSphere Data Interchange that an electronic data interchange (EDI) map developer needs to understand.
14.1 The system view of WebSphere Data Interchange

When installing and configuring a WebSphere Data Interchange environment, you need to make a distinction between a development environment and a run-time environment. In a development environment, users create maps and profiles. In a run-time environment, the translation and distribution of EDI messages occur. The platform of the run time may be different from the platform of the development environment.

14.1.1 Development environment

The development environment consists of the WebSphere Data Interchange Client on a number of Windows machines and a DB2 database accessed via Open Database Connectivity (ODBC), as shown in Figure 14-1. The database can be local or remote. It can even be on a non-Windows platform, such as AIX.

Development machines that need access to the WebSphere Data Interchange database need the DB2 Connect™ product to connect to the database.

The WebSphere Data Interchange client environment is also used to configure several aspects of the run-time environment that are not directly related to
building maps. Such settings as partner profiles, queue profiles, and locations and names of files are all set via the WebSphere Data Interchange Client. This functionality of the client may have an impact on an installation and how database access and security is configured.

When the WebSphere Data Interchange Server is using WebSphere MQ queues for input and output, it may also be required that WebSphere Data Interchange developers have an WebSphere MQ client connection to the queue manager that is used by the WebSphere Data Interchange server.

Given the possibilities for using the WebSphere Data Interchange Client, the installation and configuration of a WebSphere Data Interchange development environment involve configuration work at the database level and possibly at the WebSphere MQ level.

14.1.2 Run-time environment

The run-time environment for a WebSphere Data Interchange solution can be quite broad. There may have many possible combinations of interactions between WebSphere Data Interchange itself and other applications, as shown in Figure 14-2. But in general, WebSphere Data Interchange operates in two modes.

The first mode is the batch mode, where an automation product launches the ediservr program. This program receives instructions from a command file that contains PERFORM and other commands. The WebSphere Data Interchange Server executes these commands, which usually result in sending, receiving, and translating a number of EDI documents that are ready to process. One reason for doing this in a batch mode is to save costs. Combining several EDI documents into a single EDI transmission document is often cheaper to send over a value-added network (VAN) than sending each individual document separately.

In the second mode, when EDI documents are sent over the Internet, there is no reason to delay transmission. Communication costs for the Internet are normally not measured on a transaction basis. Therefore, a company that uses an Internet connection for EDI communication wants to send EDI documents as soon as they become available. In this situation, the second method of using WebSphere Data Interchange is launching the WDIAdapter program. This program is designed to be started by WebSphere MQ’s trigger monitor.
As shown in Figure 14-2, when a message arrives in queue from a back-end application, the trigger monitor starts the WDIAdapter program, which translates it according to the definitions in the WebSphere Data Interchange database. The translated message is then written on another queue for transmission by an Internet gateway product, such as WebSphere BI Connect.

Figure 14-2  WebSphere Data Interchange run-time environment

Given the possible scenarios in a run-time environment, the configuration and installation of the WebSphere Data Interchange Server include database tasks and WebSphere MQ tasks. Since WebSphere Data Interchange runs on a separate server, you must set up MQ communication between the wbicwdi
WebSphere Data Interchange Server components are available on Windows, AIX, and z/OS platforms. This redbook focuses on the use of WebSphere Data Interchange on Windows platforms for both the server and client environment.

### 14.2 Step-by-step implementation

This section explains how to implement both WebSphere Data Interchange server and client components so that you can use them for integration with WebSphere BI Connect.

#### 14.2.1 WebSphere Data Interchange Server installation and setup

WebSphere Data Interchange Server has an InstallShield wizard that guides you through the installation process. To use the InstallShield wizard, you must be logged in as an administrator. By default, WebSphere Data Interchange Server is installed to the C:\WDIServer32 directory.

Follow these steps to install the WebSphere Data Interchange Server:

1. Insert the WebSphere Data Interchange Server CD into the CD-ROM drive.
   a. On the menu bar, click **Start → Run**.
   b. Find the directory that contains the wdi.exe executable.
   c. Run the wdi.exe executable to start the InstallShield wizard.
2. The Welcome window opens as the InstallShield wizard prepares to install WebSphere Data Interchange Server. Click **Next**.
3. The license agreement opens. Read the information and license terms on the panel. Click the appropriate button to accept the terms of the license agreement and to indicate that you have read the notice and agree to its terms. Click **Next**.
4. Confirm that you have purchased sufficient processor license units for the number of processors you have on your computer. Provide an installation directory and click **Next** (Figure 14-3).

![Figure 14-3 Installation folder for WebSphere Data Interchange MP V3.2](image)

5. The Installer window displays a listing of the installation directory and the total size requirement. Click **Next**.

6. The InstallShield wizard begins copying program files. To stop this process at any point, click **Cancel**. The window displays the successful installation message. Click **Finish**.

The installation of the files is complete.

For the scenarios discussed in the next chapter, install CSD 12 or later for WebSphere Data Interchange Server. You can download this package from:


**Note:** Installing a CSD before you create the database does not require any special action. However, if the database exists, and depending on the CSD that you have installed before installing CSD 12, you may need to perform additional setup tasks to update the database schema. Refer to the readme file for the CSD that you are installing.
14.2.2 Setting up the WebSphere Data Interchange database

Perform the following steps while logged as a user with DB2 administrator authority. Throughout these steps, replace ediec32e with the name of the database for use by WebSphere Data Interchange.

1. If you are installing on Windows, log on with your DB2 administrator user ID.
   Click Start → Programs → IBM DB2 → Command Window to open a DB2 command window.

2. Change to the ddl directory under the installation directory. Issue the following command:
   
   db2 create db ediec32e

   When this process has successfully completed, the database has been built.

3. Enter the following command:
   
   altrec32

   This process alters some of the default parameters related to log file size and to the number of primary and secondary logs. If you choose a different name for the database, you need to edit this command file.

4. Change to the DB2 directory that contains the bind files for the DB2 utilities. Typically this is C:\Program Files\SQLLIB\bnd.
   
   For more information, see the “Binding Database Utilities” section in the appropriate DB2 Quick Beginnings book.

5. Enter the following commands:
   
   db2 connect to ediec32e
   db2 bind @db2ubind.lst messages bind.msg grant public
   db2 bind @db2cli.lst messages clibind.msg grant public
   db2 connect reset

6. Change back to the ddl directory under the installation directory.

7. Enter the following command:
   
   db2 –tf ediec32.ddl –l ec32.log

   This creates the WebSphere Data Interchange tables, indexes, views, and so on.

8. Invoke the command to issue the GRANT statements that are required to grant access to the newly created tables for the WebSphere Data Interchange Client. By default, this grants access to PUBLIC. You may want to change PUBLIC to specific user IDs or a group of authorized users:
   
   db2 –tf grntec32.ddl –l grntec32.log

   You must update the files ediec32.dll and grntec32.dll if you have chosen a different name for the database.
9. Change to the ixf directory under the installation directory.

10. Enter the following command:

```
loadec32
```

This process loads initial data into the DB2 tables. The loading may generate warnings, but you can safely ignore these. Here too, you may need to update the script to refer to the database name that you have chosen.

11. Change to the bind directory under the installation directory.

12. Invoke the command to bind the WebSphere Data Interchange DB2 packages and issue the GRANT statements that are required to grant access to the newly created tables for the WebSphere Data Interchange Server. The default in bindgrnt.fil grants access to PUBLIC. You may want to change PUBLIC to specific user IDs or a group of authorized users:

```
db2 –tf bindgrnt.fil –l bind.log
```

The file bindgrnt.fil refers again to the default database name EDIEC32E.

13. Set the run-time parameters in the properties file.

   a. WebSphere Data Interchange looks for the wdi.properties file in its current directory to find the name, user ID, and password to access the database. A sample properties file is stored in the folder samples in the WebSphere Data Interchange installation folder. Copy this sample file to the folder runtime\dicmd in the WebSphere Data Interchange installation folder.

   b. We set up this folder as the current directory when WebSphere Data Interchange starts. Edit this properties file so that WebSphere Data Interchange is using the correct queue manager and database.

You have now completed the setup of the WebSphere Data Interchange database.

### 14.2.3 Setting up WebSphere MQ objects

WebSphere Data Interchange can read XML and electronic data interchange (EDI) documents from files or MQ queues and write the translated documents back to a file or a queue. Given that the use of WebSphere MQ allows better integration with other products, such as WebSphere BI Connect or internal applications in general, we have built our environment to use queues as the input and output mechanism for WebSphere Data Interchange.

WebSphere Data Interchange provides a sample set of WebSphere MQ definitions for queues and processes. But first you must create a queue manager on the server that is used by WebSphere Data Interchange.
Perform the following steps as a user with WebSphere MQ administrator authority.

1. Select **Start → Programs → IBM WebSphere MQ → WebSphere MQ Explorer**.

2. When the MQ Explorer launches, right-click the folder **Queue Managers** and select **New → Queue Manager**.

3. In the Create Queue Manager window (Figure 14-4), provide a name for the queue manager, such as `wdi.queue.manager`, and mark it as the default queue manager if that fits your requirements. Click **Next**.

4. In the steps that follow, you can accept all default values.

5. In the panel that indicates Step 4, create a TCP/IP listener.

6. Click **Finish**.

7. We can now define the default MQ objects that WebSphere Data Interchange uses. From the command prompt change the directory to the WDI samples directory. The configuration script `wdimqcommands.txt` file is located there.

8. From the command line, enter:
   ```bash
   runmqsc < wdimqcommands.txt
   ```
If the queue manager is not set up as the default queue manager, then use its name as the first parameter:

runmqsc wdi.queue.manager < wdimqcommands.txt

The following WebSphere MQ objects are created:

- WDI.PROC: WDI trigger process definition
- WDI.FAILURE.Q: Failure queue
- WDI.INIT.Q: Trigger Initiation Queue
- ADF_IN: Application data queue for WDI translation
- EDI_IN: EDI data queue for WDI translation
- XML_IN: XML data queue for WDI translation
- ADF_OUT: Application data queue from WDI translation
- EDI_OUT: EDI data queue from WDI translation
- XML_OUT: XML data queue from WDI translation

Example 14-1 shows the definition of the process. The USERDATA parameter is used to set the current directory for the WDIAdapter program. This is the directory where the program looks for the wdi.properties file that is used to control database access.

To start the trigger monitor, run the following command from a command prompt:

runmqtrm -q WDI.INIT.Q

If the queue manager is not set up as the default queue manager, then use its name as the first parameter:

runmqtrm -m wdi.queue.manager -q WDI.INIT.Q

**Example 14-1  Process definition for WDIAdapter program**

```
DEFINE PROCESS('WDI.PROC') REPLACE +
    DESCR('WebSphere Data Interchange Adapter') +
    APPLICID('WDIAdapter.exe') +
    APPLTYPE(DEF) +
    USERDATA('C:\WDIServer32\runtime\dicmd') +
    ENVRDATA('>> "C:\WDIServer32\runtime\prt\WDIAdapter.trace"')
```

In addition to running the trigger monitor in a command window, you can add it to the WebSphere MQ service.

1. Select **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Services**.
2. When the application starts, expand the tree structure in the left pane. Right-click the queue manager used by WebSphere Data Interchange, **wdi.queue.manager**, and select **New** → **Trigger Monitor**.
3. In the Create Trigger Monitor Service window (Figure 14-5), click the **Parameters** tab.

   a. For **Queue Name**, provide the name of the initiation queue, which is by default **WDI.INIT.Q**. This name matches the sample WebSphere MQ definitions provided by WebSphere Data Interchange.

   b. Click **OK**.

   ![Create Trigger Monitor Service Window](image)

   **Figure 14-5** Creating a trigger monitor service component

This new component is not yet started, but it will start automatically after you restart the WebSphere MQ service.

During testing of WebSphere Data Interchange maps, it may be an advantage to run the trigger monitor in a command window. When the maps are tested, you can turn on the trigger monitor service component and run it in the background, in the same way that the queue manager operates.

To run the trigger monitor in a command window, use the following command:

```
runmqtrm -m wdi.queue.manager -q WDI.INIT.Q
```
14.2.4 WebSphere Data Interchange Client installation and setup

WebSphere Data Interchange Client is a Windows-based interface for WebSphere Data Interchange servers. This interface allows you to use a PC to create and maintain WebSphere Data Interchange profiles, EDI standards, data formats, and maps. It also can be used to load XML document type definitions (DTDs) and schemas into WebSphere Data Interchange.

**Note:** Support for XML schemas has been added to the product via CSD9 for the server component. The equivalent CSD for the client component is required to exploit the server support for schemas.

WebSphere Data Interchange Client has an Install wizard that guides you through the installation. As with any installation, begin by closing any applications that you have running.

1. Insert the WebSphere Data Interchange Client Installation CD-ROM. The installation process should start automatically. If it does not:
   a. On the menu, click **Start → Run**.
   b. The Run window opens with the cursor in the Open field. In the Open field, type `x:\client \WDIClient32.exe` (where `x` indicates your CD-ROM drive), and click **OK**.

2. The Welcome window displays as the Install wizard prepares to install WebSphere Data Interchange Client. Click **Next**.

3. A second Welcome window displays with messages warning you to exit all Windows programs if you have not done so already. This window also displays the license agreement and copyright. Click **Next**.

4. Read the copyright notice that displays. Click **Yes** to indicate you have read the notice and agree to its terms. Click **Next**.

   If this is the first time you are installing WebSphere Data Interchange Client, the Choose Destination Location window opens. Select your installation location by clicking **Browse** and choosing the appropriate drive and directory. You can select the drive from the drop-down list at the bottom of the window. The default destination directory is `C:\Program Files \IBM \WDI Client V3.2`.

**Important:** If you have installed previous versions of WebSphere Data Interchange Client or DataInterchange Client, you must install this version in a different directory. Click **Next**.
5. The Setup Type window opens. Choose the type of setup you want by selecting the appropriate option from the following choices:

- **Typical**: Select this option the first time you install WebSphere Data Interchange Client 3.2. This option installs all the common options and creates the software’s database. This is the default setup type.

  If you choose Typical, click **Next** and go to step 6.

**Important**: If you are reinstalling WebSphere Data Interchange Client and you select this option, you receive a warning that the Install program will overwrite your database. The Install program only overwrites the default 3.2 database files installed through a previous 3.2 install. This option does not overwrite databases installed on a user’s database system, such as DB2.

- **Custom**: Select this option when you want to choose the options to install. We recommend this option for advanced users. If you are reinstalling WebSphere Data Interchange Client, use the Custom setup to avoid overwriting your databases and drivers with the defaults.

  If you choose Custom, click **Next**. The Select Components window opens. You can choose to install one or more of the following items:

  - Program files
  - Crystal Report files
  - Database files

  Select a component by clicking the check box next to it. Click **Next**.

6. The Installer window displays a listing of the selected features and the total size required to install them. Click **Next**.

7. The Install wizard begins copying program files. To stop this process at any point, click **Cancel**. When you see a message indicating a successful installation, click **Next**.

8. The readme file is displayed. Click **Finish**.

9. Select **WDI Client** from the program folder specified during installation to start WebSphere Data Interchange Client.

**Note**: For the scenarios discussed in the next chapter, you must install CSD 8 or later for WebSphere Data Interchange Client. You can download this package from:

14.2.5 Setting up a connection to the database

You can use a WebSphere Data Interchange Client installation to interact with many databases, for example, the development and the production system. Each of these systems that you want to use has to be identified to WebSphere Data Interchange Client.

1. Start WebSphere Data Interchange Client. From the menu, click View → Systems. You see a list of defined systems, mapping to existing databases. For a new installation, you typically see the default development database, called WDIClient32DEV, which is not a DB2 database but a Microsoft Access database.

2. Create a new EDI system that uses the EDI database. From the menu, click File → New, or click the Create New Document button on the toolbar.

3. An EDI System window opens.
   a. Specify a name for the new system.
   b. Select the Data Source Name for the database.
   c. Specify the database qualifier for the database. The qualifier is sometimes referred to as the schema. The default value for the database is EDIENU32. Confirm this value with your system administrator.
   d. Identify the platform on which the WebSphere Data Interchange Server runs.
   e. Select a color to associate with this EDI System. Associating unique colors with each EDI System helps the user to easily identify which system they are working with in a multi-system environment.
   f. Click OK. The new system is saved (Figure 14-6).

---

**Figure 14-6 Creating a new system in WebSphere Data Interchange Client**
4. Exit and restart WebSphere Data Interchange Client. Upon restart, you should be able to select the new system in the drop-down box in the toolbar of WebSphere Data Interchange.

5. Click the **Setup** icon or another icon in the toolbar that results in a database access.

6. In the Connect To DB2 Database, enter the user ID and password for the run-time database, if requested. Click **OK**.

   If the Setup Functional Area window opens, the connection to your databases is complete. If there is an error, ensure that your EDI system is set up correctly. An incorrect database qualifier is a common problem (Figure 14-7).

   ![Connect To DB2 Database](image)

   **Figure 14-7** Connecting to the database

### 14.3 The component view of WebSphere Data Interchange

You should become familiar with certain concepts before you attempt to understand how a message is processed by WebSphere Data Interchange. The components of particular relevance are:

- Mailbox profiles
- Network profiles
- WebSphere MQ related artifacts
14.3.1 Mailbox profiles

Mailbox profiles contain information that WebSphere Data Interchange needs to identify the individuals and groups in your organization that receive documents to be translated. Each individual or group requires its own Mailbox profile. Figure 14-8 illustrates the default Mailbox profiles shipped with WebSphere Data Interchange.

Figure 14-8 Mailbox configuration window

Note: WebSphere Data Interchange sometimes refers to MQSeries, which is the former name of WebSphere MQ. Within the context of WebSphere Data Interchange, both names refer to the same product. MQSeries changed to WebSphere MQ starting with the version 5.3.
Of particular importance in the Mailbox profile settings are the Network ID and Receive File details. The Network ID identifies which logical network within WebSphere Data Interchange to use to send or receive information. The Network ID is selected from the list of available Network profiles available in WebSphere Data Interchange. The Receive File field defines the logical file name expected to be received by this Mailbox profile.

A mailbox can be something logical, referring to a file or an MQ queue, or referring to a mailbox provided by a VAN. In that case, the attributes Account ID, User ID, and Password are the actual account ID, user ID, and password associated with your VAN account.

Figure 14-9 illustrates the default settings of the XML_IN Mailbox profile. You see that the selected Network ID is XML, and Receive File is set to XML_IN. When using MQSeries as the communication between trading partners or applications, all other details on this window are unused. The Account ID, User ID, Password, and Msg User Class fields come into play when required by the network, for example, when using IBM Information Exchange.

14.3.2 Network profiles

Network profiles define for WebSphere Data Interchange the characteristics of the networks you use for communications with trading partners. WebSphere Data Interchange is shipped with the Network profiles required to communicate with several major networks. Figure 14-10 shows the default Network profiles shipped with WebSphere Data Interchange.
If you look at the XML profile in more detail, you can examine the details of importance. Figure 14-11 shows the default details of the XML Network profile.

Figure 14-11  Settings of a Network profile
Network program EDIRFH2 is specified along with communications routine VANIMQ, so the VANIMQ program is called to actually process the MQSeries queue specified by the Network Parameters. This network program and communications routine are shipped with WebSphere Data Interchange and are used when processing from an MQ queue. The network program EDIRFH2 is used when you expect to process or generate an MQRFH2 header.

If you do not have this requirement and you plan to use standard MQ messages, use the network program EDIMQSR. There is a third MQ-oriented network program, called EDICYCL, which is used when interacting with JMS clients. You can find examples of these different network programs for WebSphere MQ in Chapter 15, “Integration with WebSphere Data Interchange” on page 419.

The logical names of the inbound and outbound queues to be processed are kept in the field named Network Parameters. The format of the field is:

SENDMQ = name_of_the_queue_for_outbound_data RECEIVEMQ = name_of_the_queue_for_inbound_data

The actual details of the queue, such as the queue manager name, full queue name, whether destructive MQGET operations should be done, and so on, are specified in a different profile called the MQSeries® queue profile. Every WebSphere MQ queue that WebSphere Data Interchange accesses must be described within WebSphere Data Interchange by an MQSeries Queue profile.

The Envelope File field is an optional one. When translating documents, the output documents are written to this file, here XML_IN. Documents are then read from this file when sending to a trading partner. When receiving documents from a trading partner, the documents are written to this file. The WebSphere Data Interchange reads the documents from this file to translate them.
14.3.3 WebSphere MQ-related artifacts

Of particular importance to the user is an understanding of the MQSeries-related artifacts that exist in WebSphere Data Interchange and how they relate to MQ concepts such as queues and queue managers. You can find the interface with each of these MQ-related artifacts in the Setup area of the WebSphere Data Interchange client. The MQSeries Queues tab, as shown in Figure 14-12, contains WebSphere Data Interchange’s definitions of actual queues defined in WebSphere MQ.

![Figure 14-12  MQSeries Queue profiles](image-url)
When we look at the details of one of these queue definitions, we can see that WebSphere Data Interchange has assigned a logical queue name to the physical WebSphere MQ queue and defined the queue manager where this queue resides (Figure 14-13).

![Figure 14-13  Details of an MQSeries Queue profile](image)

In Figure 14-13, the Queue Profile ID XML_IN corresponds to Full Queue Name XML_IN (the actual WebSphere MQ queue). The Queue Manager field is not specified, because this queue resides on the default queue manager. If the queue is not on the default queue manager, then you specify the name of the queue manager in the Queue Manager field.

WebSphere Data Interchange uses these MQSeries Queue profiles once called from the Network Parameters field in the Network profile.
14.3.4 Service profiles

The purpose of the Service profile is to allow you to enter a utility command and all the files to use during execution of that command. There are specific fields for fixed names, such as the print file (PRTFILE), and short name and long name pairs for times when both the short and long names are user defined, such as input and output files. Figure 14-14 shows the default Service profiles shipped with WebSphere Data Interchange.

Figure 14-14 Default Service profiles
In the General tab of the XML_IN Service profile (Figure 14-15), you can see the default command provided by WebSphere Data Interchange.

The XML_IN parameters passed to INFILE tell WebSphere Data Interchange that this is the file to perform translation on (to TRANSFORM). This corresponds to the Receive File detail that we specified in the Mailbox profile. The X parameter passed to SYNTAX tells WebSphere Data Interchange to expect XML_IN to be in an XML format.

![Figure 14-15 Detailed view of the settings of a Service profile](image)

*Figure 14-15  Detailed view of the settings of a Service profile*
The Common Files tab outlines the default locations for each of the file structures used by WebSphere Data Interchange to provide the user with information about the TRANSFORM process (Figure 14-16). For more details, see Chapter 15, “Integration with WebSphere Data Interchange” on page 419.

The Input Files tab is typically left blank. This is used only if the input for the command is different from the file that initially triggered the process. This is not a common scenario.

The Output Files tab associates a physical file location for the logical output of the utility command. For example, enter the following command on the General tab:

```
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X) OUTFILE(XML_OUT)
```
The output of the utility command resides in the physical file location associated with XML_OUT in the Output Files tab (Figure 14-17). In Figure 14-17, you see that the logical file name XML_OUT is mapped to the physical file name ..\xml\xml_out.txt.

![Figure 14-17   Output Files tab of a Service profile](image)

The name you give to the Service profile is its logical name. If another command writes information to the file associated with this logical name, the PERFORM command is executed after that command completes, connecting the commands together. This is known as command chaining.

The Network Files tab allows the user to enter details of files required for communication by the network program. Typically these are used if Expedite is used as the communication channel between WebSphere Data Interchange and Information Exchange. If using WebSphere MQ-to-WebSphere MQ communications, these fields can remain unused.
14.3.5 Trading Partner profiles

Trading Partner profiles are maintained in WebSphere Data Interchange under the Trading Partner icon (Figure 14-18).

WebSphere Data Interchange ships with two sample Trading Partner profiles (Figure 14-19). The ANY trading partner is a useful template that can be used when simulating a data transformation scenario where the sender and receiver details are of little importance.

![Image](Figure 14-19  Default Trading Partner profiles)
Figure 14-20 shows the General tab of the ANY Trading Partner profile.

In a simple ANY to ANY scenario over WebSphere MQ, the only field of importance is the Network ID field. This determines which WebSphere Data Interchange Network profile to use with this trading partner. This should correspond with the network ID selected in the Mailbox profile as illustrated earlier. If Information Exchange is being used, the user is required to enter an Information Exchange Account and User ID. It is also necessary to identify the Interchange Attributes by entering the Trading Partner's Qualifier and ID. The ID in this instance can be an alias, depending on what you define in Information Exchange.

In WebSphere Data Interchange's view of the world, there are two type of trading partners: application (or internal) trading partners and EDI (or external) trading partners. An application trading partner represents a business entity within the customer's enterprise. An external trading partner is a business entity that the user's enterprise does business with via EDI. Both are represented by a Trading Partner profile. What differentiates the two of them is the trading partner type field on the General tab of the Trading Partner Profile editor. In Figure 14-20, you see that ANY is defined as both an EDI and an application trading partner.
WebSphere Data Interchange provides tabs to store more specific details on the trading partner. For example, space is provided for company information and contacts. These are not required, however.

The only other tab of importance in this setup is the WDI Proc Options tab. Here the user is allowed to specify the delimiters used by WebSphere Data Interchange. Figure 14-21 shows the default settings of ANY.
14.3.6 Rules

How does WebSphere Data Interchange know which map to use and on which queue to put the output data? WebSphere Data Interchange makes this decision based on the usage (rules) defined by the user for a particular map. This area is known as “rules” in WebSphere Data Interchange. Conceptually it is simplest to think that whenever a message arrives on a queue monitored by WebSphere Data Interchange, there is a “rule” (created by the user) that defines what WebSphere Data Interchange should do with that message.

Each rule relates to a particular map. Therefore you must create a map first before you can add a rule to determine its usage. You can see all maps in your system by clicking the map icon (top icon in Figure 14-22).

If you click the map to select it, you can either see existing usages or create new usages by clicking the usage icon (bottom icon in Figure 14-22).

Figure 14-23 shows the sample data transformation map shipped with WebSphere Data Interchange. Notice that it has also been shipped with a sample rule for usage.

![Figure 14-23 Sample rule for a data transformation map](image)
Looking at the detail of this rule, you can identify the areas of importance (Figure 14-24).

The Map Name, Dictionary Name and Document Name are all in grey and cannot be changed, since they are pulled from the map detail. In the Associated With frame, the user has the opportunity to associate the map with either a particular process or a set of Trading Partners. If selected, the process field allows the map to become associated with a particular business process, for example 850, 855, and so on. If the Trading Partner field is selected, the user has the opportunity to specify a sender or receiver. These drop-down menus are populated by the list of trading partners available in the Trading Partner profile area described earlier.

To activate a rule to allow a map to perform data transformation, select Active in the Properties frame and set the Usage Indicator to Production. You can also specify an output file name and type here. This detail in this field is overwritten if an output file is specified in the PERFORM statement of the Service profile as described earlier.
The Envelope Attributes tab defines the type of envelope to be used on the EDI message after it is created. Figure 14-25 shows that X has been selected. X in this instance defines an ANSI X12 standard envelope.

![Envelope Attributes tab of the usage of a map](image)

Figure 14-25   Envelope Attributes tab of the usage of a map

The WDI Options tab allows the user to specify varying levels of validation on a translation taking place using this usage.

### 14.3.7 Final view of a running WebSphere Data Interchange system

Follow this example of a typical flow:

1. An application sends an XML message to WebSphere Data Interchange via WebSphere MQ to be translated to EDI.
2. Using MQPUT, the application places the XML message into the XML_IN queue.
3. WebSphere Data Interchange adapter is triggered by the WebSphere MQ trigger monitor. When triggered, the adapter:
   a. Reads the wdi.properties file for run-time parameters
   b. Initializes WebSphere Data Interchange
      
      If WebSphere Data Interchange cannot be initialized, the adapter turns off triggering for the queue and then terminates.
c. Sets the name of the file that the message will be received into, which is `datadirectory` (from property file), `rcvdirectory` (from property file), `MQSeries message ID` (from MQMD).rcv

d. Browses the data queue to find information about the next available message

e. If it is OK to proceed, calls WebSphere Data Interchange with:

```
PERFORM RECEIVE AND PROCESS WHERE REQID(mq_queue_name[1-16])
       BATCHSET(batchid)
```

4. WebSphere Data Interchange retrieves the Mailbox profile.

a. The short name of the file to receive into is determined from the Receive File field.

b. WebSphere Data Interchange determines the Network ID of this Mailbox profile and goes to the relevant Network profile to determine the queue to receive from.

5. WebSphere Data Interchange retrieves the Network profile.

a. WebSphere Data Interchange determines the Network Program and Communications Routine to actually process the queue.

b. The Network Parameters specify the MQSeries Queue profiles to be used when sending and receiving over WebSphere MQ.

6. WebSphere Data Interchange retrieves the MQSeries Queue profile specified by the RECEIVEMQ statement in the Network Parameters:

   The physical WebSphere MQ queue name is retrieved along with the relevant Queue Manager (if no queue manager is specified, the default queue manager is used).

7. WebSphere Data Interchange retrieves the message and writes the body to a file using the short name specified in the Mailbox profile. A destructive MQGET is then performed under syncpoint control.

8. The PERFORM RECEIVE command completes but notices that data was written to the Receive File specified in the Mailbox profile. WebSphere Data Interchange then looks for a Service profile of the same name.

9. The corresponding Service profile is retrieved. The next entry in the command chain begins executing. This is taken from the PERFORM Command area of the Service profile. Typically this command takes the following format:

```
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X)
```
10. WebSphere Data Interchange attempts to retrieve an active map rule that corresponds to this TRANSFORM type. The rule is identified based on the DTD received and the combination of trading partners and processes identified. The rule specifies the output of the TRANSFORM command, for example, EDI_OUT.

11. The PERFORM TRANSFORM command completes. WebSphere Data Interchange sees that data was written to the output file specified in the rule, and looks for a Service profile with a matching name.

12. Again, the next entry in the command chain is retrieved from the PERFORM Command area of the Service profile. Typically, this command takes the format:

   PERFORM SEND WHERE REQID(EDI_OUT) FILEID(EDI_OUT) CLEARFILE(Y)

13. WebSphere Data Interchange retrieves the mailbox ID as specified by the REQID parameters, gathers the Network ID, and examines the corresponding Network profile. The Network Parameters here define the MQSeries Queue profile to be used by the SENDMQ statement. WebSphere Data Interchange sends EDI_OUT (as specified by the FILEID parameters) to the queue as defined in the MQSeries Queue profile. When sent, the file is cleared.

14. The command is now complete. There is nothing on the next command chain, so the utility returns to the MQSeries adapter.

15. The adapter program calls the msgTransform exit (if present) with the return codes from the translation. The exit returns SYNC_CONTINUE, the adapter takes a syncpoint, and then starts over.
Integration with WebSphere Data Interchange

This chapter discusses several scenarios of routing and transformation based on the document type and destination or origin. It demonstrates how WebSphere Data Interchange and WebSphere BI Connect work together to provide a solution for routing and transformation by hiding these complexities in the middleware layer and not in the application.
15.1 Overview of implemented scenarios

The different scenarios extend the infrastructure that we used previously in this redbook. Before we explain how to implement the scenarios, let’s examine some of the details about these scenarios.

15.1.1 Phase 1: Transforming XML into EDI for Company A

In the first scenario of this chapter, the WebSphere BI Connect Enterprise server of Company E uses WebSphere Data Interchange to perform data translation services. Electronic data interchange (EDI) transactions that are received from the WebSphere BI Connect Advanced of Company A are routed to a Java Message Service (JMS) queue, which points to the WebSphere Data Interchange server. The WebSphere Data Interchange server reads the messages received from the WebSphere BI Connect Enterprise server and translates them accordingly into an XML format.

A similar flow exists for the outbound case. XML documents generated by a back-end application are sent to a WebSphere MQ queue and retrieved by WebSphere Data Interchange. This then translates them to the appropriate EDI format and sends them to WebSphere BI Connect Enterprise via WebSphere MQ.

Starting with the WebSphere BI Connect environment that we have built and used throughout this redbook, you must perform several steps. Since the WebSphere BI Connect server of Company E has interacted so far only with the file system, you must now perform steps so that JMS queues become a valid gateway and target. Since messages need to flow between the queue manager of WebSphere BI Connect and the queue manager of WebSphere Data Interchange, you must also configure the two queue managers, so that they can exchange messages.

When the WebSphere MQ system works as expected and messages can be exchanged, you must configure the JMS layer on top of WebSphere MQ. The JMS objects are created using the JMSAdmin tool that is part of WebSphere MQ. At that time, we can set up the JMS gateway and the JMS target.

Now that you can exchange MQ and JMS messages between the WebSphere Data Interchange server and the WebSphere BI Connect server, you can develop the translation maps between EDI and XML. The XML format that is in use for Company E, is a custom format XML defined by a document type definition (DTD), which you import in the WebSphere Data Interchange repository.

When the maps are developed and rules are created, you can perform a validation of this first scenario (Figure 15-1).
15.1.2 Phase 2: EDI for Company A and XML for Company X

In this scenario, again you add Company X to the picture. The back-end application still generates the same type of XML documents. However, the XML documents for Company A need to be transformed, while the XML documents do not need any special transformation.

The most obvious solution may be to adjust the back-end application so that messages destined for Company X are sent directly to WebSphere BI Connect, thus bypassing the WebSphere Data Interchange server. However, this means that the B2B capabilities of Company A and Company X become embedded in the application logic. From a business perspective, the back-end application should be able to drop the same type of XML messages for both partners in the same queue and the B2B solution (WebSphere BI Connect, WebSphere Data Interchange, or both) should handle it from there.

That is exactly what you are going to perform. You develop a main map in WebSphere Data Interchange that is called for each XML document. This map inspects the business identifiers in the XML document. Based on the value of that business identifier, Company X or Company A, the main map performs a context switch and passes control to the appropriate map: an XML-to-XML map for Company X and an XML-to-EDI map for Company A (Figure 15-2).
15.1.3 Phase 3: Managing business identifiers

In the previous phase, we used business identifiers in three different locations. The XML document generated by the back-end application contained a business identifier for the sender and receiver of the document. This value is also defined in WebSphere Data Interchange in a trading partner profile and is used in the map to perform some conditional logic. Finally, business identifiers are used by WebSphere BI Connect to look up connection information. And, you can add a fourth location where these business identifiers are defined: the B2B gateway of the target company.

In reality, it happens quite often that the management and synchronization of these business identifiers is disconnected. It is unlikely that Company A will be known to Company E as companya, which is the business identifier of Company A that we configured in WebSphere BI Connect and in WebSphere Data Interchange.

Assume for a moment that Company A is known as TP_A in the back-end applications of Company E. To solve this mismatch of business identifiers, you can create a lookup table in WebSphere Data Interchange for these identifiers. By adding a simple Translate() function in the transformation map, WebSphere Data Interchange replaces TP_A in the document with the correct business identifier companya.
15.1.4 Phase 4: Multiple EDI transactions and internal applications

Phase 2 and phase 3 consist of routing scenarios for the outbound flow. Routing is based on destination, not on document type, which we have limited to EDI 850 and its equivalent XML document.

Assume now for a moment that you need to cater for more than purchase order transactions. As an example, consider an inventory inquiry transaction (EDI 846). It is common that inventory management and sales management are performed by a different application. When WebSphere Data Interchange translates the incoming EDI 846 and 850 documents, it should send the equivalent XML documents to different destinations. Thus, you have one queue as input to WebSphere Data Interchange containing multiple types of documents and multiple output queues, one for each document type.

This filtering may sound complex, but the actual implementation in WebSphere Data Interchange is rather simple (Figure 15-3).

![Diagram of routing incoming documents based on document type](image-url)
15.2 Setting up JMS for WebSphere BI Connect

This section discusses the setup of JMS for use by WebSphere BI Connect. It begins with the setup of WebSphere MQ itself so that messages can be exchanged between the two queue managers. Next, it defines the mapping between MQ resources and JMS resources. These JMS resources are then used when defining a JMS gateway and JMS target in WebSphere BI Connect.

15.2.1 Connecting the queue managers

The infrastructure of Company E consists of three individual machines, as shown in Figure 15-1 on page 421. To make sure that messages can flow from the WebSphere Data Interchange to the WebSphere BI Connect server via the queue manager that is used by WebSphere BI Connect, you must create a number of MQ objects to support this intercommunication.

While we do not discuss all aspects of MQ networks, note that two types of techniques exist:

- Classic MQ intercommunication via sender/receiver channels and transmission queues
- Clustered MQ communication via cluster channels and a shared transmission queue

For a two-machine setup, clustered MQ communication does not add many advantages over classic MQ intercommunication. For larger networks, MQ cluster technologies reduce the administration overhead and can help in load-balancing and high availability. For the purposes of this redbook, we use classic MQ intercommunication. The required MQ definitions can be grouped into two types. One type enables MQ intercommunication between the two queue managers. The second type consists of symbolic definitions on the local queue manager pointing to actual objects on the remote queue manager.

To enable communication between the queue managers wdi.queue.manager and partner_e.bcg.queue.manager, you must run commands on both servers.

For the queue manager partner_e.bcg.queue.manager, enter the following commands using the interactive command interface runmqsc of WebSphere MQ:

```
define q1(TO.WDI) usage(XMITQ) trigger trigtype(FIRST)
initq(SYSTEM.CHANNEL.INIQ) trigdata(TO.WDI)

define chl(TO.WDI) chltype(SDR) trptype(TCP) conname('wbciwdi(1414)')

define chl(TO.WBIC) chltype(RCVR)
```
For the queue manager wdi.queue.manager, enter the following commands using the interactive command interface **runmqsc** of WebSphere MQ:

```
define ql(TO.WBIC) usage(XMITQ) trigger trigtype(FIRST)
  initq(SYSTEM.CHANNEL.INIQ) trigdata(TO.WBIC)

define chl(TO.WBIC) chltype(SDR) xmitq(TO.WBIC) conname('wbicdata(9999)')
  trptype(TCP)

define chl(TO.WDI) chltype(RCVR)
```

In each block of commands, the first command creates a local transmission queue to store messages before they are sent to the remote queue manager. The last three parameters enable automatic transmission of the messages as soon as they become available. The value TO.WDI (TO.WBIC in the second block of commands) refers to the sender channel object that needs to be started when the first message arrives.

The second command defines the sender channel. Its name matches the definition of the receiver channel on the remote queue manager. It also refers to the transmission queue defined earlier. The two other parameters allow you to set communication protocol, TCP, and the host name and port. Finally, the last command defines the receiver channel.

Messages that are received by WebSphere BI Connect Enterprise need to arrive at the queue EDI_IN hosted by queue manager wdi.queue.manager. To create a symbolic link from the queue manager partner_e.bcg.queue.manager, enter the following command in a **runmqsc** session:

```
define qr(TO.EDI_IN) rname(EDI_IN) rqmname(wdi.queue.manager) xmitq(TO.WDI)
```

You can also create each of these MQ objects by using the WebSphere MQ Explorer application. Messages that are prepared by WebSphere Data Interchange for transmission to Company A must be retrieved by WebSphere BI Connect from a local queue on queue manager partner_e.bcg.queue.manager.

To create such a queue, use the following command:

```
define qlocal(EDI_OUT)
```

Finally, you must create a remote queue object on queue manager wdi.queue.manager to point to the local queue on the queue manager partner_e.bcg.queue.manager. Use a command such as the following example to create the remote queue:

```
define qr(TO.EDI_OUT) rname(EDI_OUT) rqmname(partner_e.bcg.queue.manager)
  xmitq(TO.WBIC)
```
Figure 15-4 summarizes the setup of both queue managers.

Note: Strictly speaking, we can create an MQ configuration whereby WebSphere BI Connect stores and retrieves messages directly in the queues owned by the queue manager wdi.queue.manager. As such, we would bypass the queue manager partner_e.bcg.queue.manager, which is used by WebSphere BI Connect for communication between individual components of WebSphere BI Connect. However, in reality, WebSphere BI Connect is located in a secured network environment. Network administrators want to limit the communication between WebSphere BI Connect and the internal network. Using the queue manager partner_e.bcg.queue.manager as an intermediate assists in limiting sessions to the internal network.
15.2.2 Enabling JMS

WebSphere BI Connect uses the JMS API to interact with WebSphere MQ. The JMS API is part of the Java 2 Platform, Enterprise Edition (J2EE), specification. It is a layer that is developed on top of the standard WebSphere MQ API. You must set up the mapping between JMS objects and WebSphere MQ objects.

The mapping is retrieved by the JMS API using the Java Naming and Directory Interface (JNDI) API. JNDI providers can be anything, for example Lightweight Directory Access Protocol (LDAP) servers or naming services provided by other WebSphere Application Server instances in your network or a simple file. The actual mapping commands are independent of the JNDI provider.

WebSphere MQ provides the JMSAdmin tool to store the links between a JMS object and a WebSphere MQ object in a JNDI directory. You set up your choice of JNDI provider in the JMSAdmin.config configuration file, which you can find in the Java\bin directory in the installation folder of WebSphere MQ. You perform the mapping on the machine where the JMS application is running, and not on the machine that runs the actual JMS server (or queue manager).

1. On the WebSphere BI Connect machine, open the file jmsadmin.config in the folder C:\WMQ\Java\bin in a text editor.

2. Uncomment the INITIAL_CONTEXT_FACTORY setting for the JNDI provider of your choice. Example 15-1 has it set for the file-based JNDI provider. Each JNDI provider needs to be addressed in a different way, which is expressed via the PROVIDER_URL parameter.

3. Update the file-based URL to point to a valid directory on the file system, for example C:\WMQ\Java\JNDI. Notice that the syntax of PROVIDER_URL uses the forward slash instead of the normal backward slash of Windows.

Note: Make sure that you uncomment only one setting of each parameter. If you want to use multiple providers, you need to create multiple configuration files and pass the name of the configuration file as a parameter when running the tool JMSAdmin.
Example 15-1  JMSAdmin configuration file

```plaintext
# The following line specifies which JNDI service provider is in use.
# It currently indicates an LDAP service provider. If a different
# service provider is used, this line should be commented out and the
# appropriate one should be uncommented.
#
#INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory
#
# The following line specifies the URL of the service provider's initial
# context. It currently refers to an LDAP root context. 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:/WMQ/Java/JNDI
#PROVIDER_URL=iiop://localhost/
```

4. You can now use the JMSAdmin tool to create the mapping between JMS objects and MQ objects.

   Start JMSAdmin in a command window, with C:\WMQ\Java\bin as the current directory.

5. Create a mapping between a queue connection factory object and a queue manager. You also need two mappings between the JMS queues for inbound and outbound messages and the MQ queues that we defined in 15.2.1, "Connecting the queue managers" on page 424.

   a. Figure 15-5 lists the commands used in JMSAdmin to set up the mapping. Use the first command to create a context, called WBIC_JMS. You can think of a context as a folder to store related objects.

      You are not required to create a context. However, it is always good practice to group common objects together. Defining all JMS objects in the root context can make things confusing and difficult to manage.

   b. After you create the context, switch to it by entering:

      chg ctx(WBIC_JMS)

   c. Create the three JMS mappings within this context.

      i. The first object is called a queue connection factory. Given that the JMS objects are used on a different machine than the machine that hosts the MQ objects, set up the queue connection factory as an MQ client object. This is reflected in the third command in Figure 15-5.
The queue connection factory object uses the client transport to talk to the existing queue manager partner_e.bcg.queue.manager. You can connect this queue manager via the MQ client channel object called `java.channel`. Furthermore, the queue connection factory object refers to the host name and the port that is used by the MQ listener component. The MQ client object `java.channel` is created as part of the WebSphere BI Connect configuration of the queue manager partner_e.bcg.queue.manager. This same object can be reused for the purposes of creating a JMS gateway.

ii. The next object is a JMS queue that points to the MQ queue called TO.EDI_IN. Remember that the queue TO.EDI_IN was created as a remote queue object, pointing to the local queue EDI_IN on the queue manager wdi.queue.manager. In the naming of the JMS object, we chose to hide this fact by choosing a name that does not imply any remote characteristic.

iii. The last object, JMS queue EDI_OUT, refers to the local queue EDI_OUT hosted by queue manager partner_e.bcg.queue.manager.

6. To end the interactive command session in JMSAdmin, use the `end` command.
15.2.3 Creating the JMS gateway

With the JMS definitions in place, you can proceed with the definition of a JMS gateway for the WebSphere BI Connect Enterprise Community Manager.

1. Log on as hubadmin to the console of Company E.
2. Select **Account Admin → Profiles → Community Participant** and click **Search**.
3. When you see the list of community participants, open the profile of Company E. Then select **Account Admin → Profiles → Gateways**.
4. You should see the list of gateways that are currently defined, which includes the FileSystemGateway defined in Chapter 7, “Creating a basic B2B exchange” on page 121. Click **Create** to add a new gateway.
5. In the new Gateway List window (Figure 15-6), complete these tasks:
   a. For Gateway Name, type a new name, such as **JMSGateway**.
   b. For Transport, select **JMS**. Now you see the new attributes that are specific for the JMS transport.
   c. For Target URI, specify the provider URL for JNDI, which is `file://C:/WMQ/Java/JNDI/WBIC_JMS` in our setup.
   d. For JMS JNDI Factory Name, specify the implementation class of the JNDI interface, which is `com.sun.jndi.fscontext.RefFSContextFactory`.
   e. For JMS Factory Name, specify the queue connection factory **WBIC_QCF**. For JMS Queue Name, specify the queue **EDI_IN**.
   f. Click **Save** to store the new gateway.

**Note:** The Target URI includes the context defined in JMSAdmin. However, this is not required. Alternatively, you can prefix the JMS Factory Name and the JMS Queue Name attributes with the context:
- JMS Queue Name: **WBIC_JMS/EDI_IN**
- JMS Factory Name: **WBIC_JMS/WBIC_QCF**
Before you can use the new JMS gateway, you need to update the participant connection that was used earlier for receiving AS2 documents from Company A. Previously, these EDI documents were stored in the edi_in directory. Now, you want to store these documents in the queue EDI_IN, represented by the JMS gateway.

1. Select **Account Admin** → **Participant Connections**.
2. Select **Company A** as the source and **Company E** as the target and click **Search**.
3. Locate the connection that has AS as the source package and None as the target package. Click **Gateways** for that connection.

4. In the Edit Participant Connection window (Figure 15-7), change the attribute Target Gateways to **JMSGateway**. Click **Save**.

![Figure 15-7 Changing the gateway on the participant connection](image)

15.2.4 Creating the JMS target

Creating a JMS gateway allows you to receive EDI documents in a JMS queue. Now you want to create a JMS target, which allows you to send JMS messages that contain EDI documents to WebSphere BI Connect. In 15.2.1, “Connecting the queue managers” on page 424, we defined the queue EDI_OUT for that purpose. Create a JMS target mapping to that queue.

1. Log on to the console of WebSphere BI Connect as hubadmin and select **Hub Admin** → **Hub Configuration** → **Targets**.
2. When you see the list of targets, click **Create Target**.
3. In the Target Details window (Figure 15-8), complete these items:
   a. For Target Name, type a new name, such as JMSTarget.
   b. For Transport, select **JMS**. More attributes appear.
c. For JMS Provider URL, specify the provider URL for JNDI, which is file://C:/WMQ/Java/JNDI/WBIC_JMS in our setup.

d. For JNDI Factory Name, specify the implementation class of the JNDI interface, which is com.sun.jndi.fscontext.RefFSContextFactory.

e. For JMS Factory Name, specify the queue connection factory WBIC_QCF.

f. For JMS Queue Name, specify the queue EDI_OUT.

g. Click **Save** to store the new target.

![Target Details](image)

**Figure 15-8 Creating a JMS target**
4. Before you can use the new JMS target, update the participant connection that was used earlier for sending EDI documents to Company A. Previously, these EDI documents were retrieved from the directory edi_out or the target called FileSystemTarget.

5. Retrieve these documents in the queue EDI_OUT, represented by the JMS target.
   a. Select **Account Admin → Participant Connections**.
   b. Select **Company E** as the source and **Company A** as the target and click **Search**.
   c. Locate the connection that has None as the source package and AS as the target package. Click **Gateways** for that connection.
   d. In the Edit Participant Connection window (Figure 15-9), change the attribute Source Gateways to **JMSTarget**. Click **Save**.

![Connection Management Gateway](image)

*Figure 15-9 Changing the gateway on the participant connection*


### 15.2.5 Validating the JMS and MQ configuration

The JMS configuration is now complete. Received EDI documents (see Figure 15-10) are now stored in a queue, and EDI documents to be sent can be retrieved from a queue as well. Before you proceed with EDI mapping in WebSphere Data Interchange, verify the configuration so far.

Drop an EDI file in the edi_out directory of Company A. If all goes well, the EDI document is received by Company E and is sent onto the remote queue TO.EDI_IN, which points to the local queue EDI_IN hosted by the queue manager wdi.queue.manager.

To verify the message contents, you can use a utility, such as RFHUTIL, which is available for free as SupportPac IH03 from the SupportPacs Web site at:


---

**Figure 15-10  Received EDI message**
Inspect the other tabs of RFHUTIL. Select the RFH tab, for example, which is shown in Figure 15-11. As expected, the RFH tab shows that this MQ message is a JMS message, due to the presence of a JMS folder in the RFH header. It also has an mcd folder, which has jms_text as the message domain and application/edi-x12 as the message type.

![Figure 15-11  Rules, formats header for a message sent by WebSphere BI Connect](image)

There is also a usr folder, for which the contents are shown in Example 15-2.

Example 15-2  Contents of usr folder for a message sent by WebSphere BI Connect

```xml
<content_type>application/edi-x12</content_type><content_length dt='i8'>1505</content_length>
```
The contents of the usr folder are greatly impacted by the choice of packaging within WebSphere BI Connect. The participant connection that is used to store this incoming EDI message in a queue has None as its target packaging. An alternative packaging method is Backend Integration. This type of packaging adds many more attributes to the usr folder. These attributes contain details about the source and target business identifiers and type of document. This can be useful for integration with other applications or integration platforms. For integration with WebSphere Data Interchange, this is not required.

Example 15-3 shows the contents of the usr folder for a message sent by WebSphere BI Connect when using the packaging option of Backend Integration.

Example 15-3  Contents of usr folder packaged for Backend Integration

```xml
<x_aux_protocol>customXML</x_aux_protocol>
<content_type>application/xml; charset=UTF-8</content_type>
<content_length dt='i8'>124</content_length>
<x_aux_third_party_bus_id>105217165</x_aux_third_party_bus_id>
<x_aux_sender_id>987654321</x_aux_sender_id>
<x_aux_process_version>1.0</x_aux_process_version>
<x_aux_protocol_version>1.0</x_aux_protocol_version>
<x_aux_system_msg_id>10837833923730000660E759003720000000000036</x_aux_system_msg_id>
<x_aux_process_type>Invoice</x_aux_process_type>
<x_aux_receiver_id>123456789</x_aux_receiver_id>
```

To test the outbound flow, use these steps:

1. Write a message to the queue TO.EDI_OUT on queue manager wdi.queue.manager.
2. Verify that it arrives at the queue EDI_OUT on queue manager partner_e.bcg.queue.manager.
3. Verify that it is picked up by WebSphere BI Connect and arrives at the intended business partner.
Again you can use the tool RFHUTIL for such a test. Figure 15-12 shows a file that is being read and then sent to the queue TO.EDI_OUT.

![Diagram](image)

*Figure 15-12  Reading from a file and writing to a queue*
Figure 15-13 shows the contents of the RFH header as it was set before sending the actual message. We selected a version 2 header and set the data format to MQSTR. We chose to include a JMS folder, without setting any of the attributes in the JMS folder. Also, we did not send a usr or mcd folder.

You now have an environment where you can use MQ messages to send and receive EDI documents to and from your partners. You also have an idea about how a message is displayed when it is processed by WebSphere BI Connect and the attributes that are required when sending a message to WebSphere BI Connect. All of this helps when configuring WebSphere Data Interchange to prepare EDI documents starting from an XML document.

15.3 Handling the inbound document flow

Starting with a base WebSphere Data Interchange installation, this section explains the steps to transform an incoming EDI 850 into a custom XML format.
15.3.1 Document definition for XML

The first step when creating a data transformation map for translating EDI to XML is to create a dictionary to store the XML DTD document. You can import XML DTD documents into the WebSphere Data Interchange database. You do this import using the client interface of WebSphere Data Interchange.

1. Start the client program and click the XML button in the tool bar.

2. When the XML Query is opened, click the Create New Document button in the tool bar, or alternatively, select File → New.

3. The XML Dictionary window (Figure 15-14) opens.
   a. Enter a name for the new dictionary to hold the DTD documents or XML schema definitions.
   b. Optionally type a suitable description for this dictionary.
   c. Click the Save button in the toolbar or select File → Save.

4. Now that you have a dictionary to hold XML definitions, import the DTD. Select File → Open Import File to start the import process.
5. In the Select Import File window, complete these tasks:
   a. Change the file type to **XML DTD file**.
   b. Browse your disk to locate and select the required DTD that describes the XML document. The other file type, export/import files (eif), is the native file format for WebSphere Data Interchange. This file format is used to import and export EDI data, such as maps, EDI standards, and even XML dictionaries.
   c. After you select the DTD, WebSphere Data Interchange prompts you for the name of the system into which you want to import this DTD (Figure 15-15). Remember that you can manage multiple instances of the WebSphere Data Interchange databases from a single interface. Select the system that you created before. Click **OK**.

![Figure 15-15  Select a System](image)

6. The Import XML DTD window (Figure 15-16) enables you to provide more details about the DTD that you are importing. Verify the name of the dictionary into which you are importing this DTD. Also provide the name of the root element. Remember that the element name is case sensitive. Click **Import**.

![Figure 15-16  Import XML DTD](image)
Example 15-4 shows the DTD that we use for this configuration.

**Example 15-4  DTD for a purchase order**

```xml
<!--Example of a Purchase Order Created in XML-->

<!ELEMENT purchaseOrder (fileHeader, orderHeader, lineDetail*, lineText*, fileTrailer)>

<!ELEMENT fileHeader
    (date,
     time)>

<!ELEMENT date (#PCDATA)>

<!ELEMENT time (#PCDATA)>

<!ELEMENT orderHeader
    (orderNumber,
     orderFlag,
     orderDate,
     contractNumber,
     supplierNumber,
     supplierId,
     supplierAddress1,
     supplierAddress2,
     supplierAddress3,
     buyerName,
     buyerSerial,
     currencyCode,
     paymentTerms,
     customerId,
     deliveryAddress1,
     deliveryAddress2)>

<!ELEMENT orderNumber (#PCDATA)>

<!ELEMENT orderFlag (#PCDATA)>

<!ELEMENT orderDate (#PCDATA)>

<!ELEMENT contractNumber (#PCDATA)>

<!ELEMENT supplierNumber (#PCDATA)>

<!ELEMENT supplierId (#PCDATA)>

<!ELEMENT supplierAddress1 (#PCDATA)>

<!ELEMENT supplierAddress2 (#PCDATA)>

<!ELEMENT supplierAddress3 (#PCDATA)>

<!ELEMENT buyerName (#PCDATA)>

<!ELEMENT buyerSerial (#PCDATA)>

<!ELEMENT currencyCode (#PCDATA)>

<!ELEMENT paymentTerms (#PCDATA)>

<!ELEMENT customerId (#PCDATA)>
```
<!ELEMENT deliveryAddress1 (#PCDATA)>
<!ELEMENT deliveryAddress2 (#PCDATA)>

<!ELEMENT lineDetail
(lineNumber,
partNumber,
itemDesc,
quantity,
 uom,
unitPrice,
dueDate)> 

<!ELEMENT lineNumber (#PCDATA)>
<!ELEMENT partNumber (#PCDATA)>
<!ELEMENT itemDesc (#PCDATA)>
<!ELEMENT quantity (#PCDATA)>
<!ELEMENT uom (#PCDATA)>
<!ELEMENT unitPrice (#PCDATA)>
<!ELEMENT dueDate (#PCDATA)>

<!ELEMENT lineText
(orderNumber,
lineNumber,
text)> 

<!ELEMENT orderNumber (#PCDATA)>
<!ELEMENT lineNumber (#PCDATA)>
<!ELEMENT text (#PCDATA)>

<!ELEMENT fileTrailer
(totalDollars,
totalQuantity)> 

<!ELEMENT totalDollars (#PCDATA)>
<!ELEMENT totalQuantity (#PCDATA)>
After your DTD import is finished, you can find your new object on the DTDs tab as shown in Figure 15-17.

![Figure 15-17 List of DTDs available in WebSphere Data Interchange database](image)

### 15.3.2 Document definition for EDI

For each EDI document standard, there are several versions and releases. You can download import files for ANSI X12 and other standards from the following Web site:


You can download each standard as an export/import file (eif) for WebSphere Data Interchange. Select **File → Open Import File** to load the definitions in your database. For our purposes, we used ANSI X12 Standard Version 4 Release 1. The 850 transaction (purchase order) is usually part of the standard installation of WebSphere Data Interchange. You must import other transactions manually after downloading the standards file from the Web site.

### 15.3.3 Transformation map for EDI to XML

Transformation maps hold the translation commands that WebSphere Data Interchange needs to execute when translating one type of document into another.

1. Open the map editor, and click the **Map** button in the tool bar. There are several types of maps that WebSphere Data Interchange can handle.

2. To translate XML into EDI, the most appropriate type of map is the data transformation map. Click the **Data Transformation Maps** tab to work with this type of map.
3. Click the **Create New Document** button on the toolbar or select **File → New** to start the definition of a new transformation map.

4. A map definition wizard opens that enables you to qualify the map. In the Create a Data Transformation Map – Map Name window (Figure 15-18), specify a name and a description for the map. Click **Next** to continue.

![Create a Data Transformation Map - Map Name](image)

*Figure 15-18  Create a Data Transformation Map: Map Name*
5. In the Create a Data Transformation Map – Source or Target window (Figure 15-19), select either Source or Target. This option controls how the actual map commands are built and executed. In this case, where we map between XML and EDI 850, the difference between Source and Target is:

- **Source map**: Given the element REF in my EDI document, to which field in my XML document should I map this?
- **Target map**: Given the element From in my XML document, from where does it get its value?

The option you choose controls the order in which elements are mapped. In the Source case, WebSphere Data Interchange traverses the XML tree and decides for each element what needs to be done with the value. In the Target case, the parser looks at each possible data element in the EDI document and looks for the value that this data element should receive. In this case, we select a **Target** map.

Click **Next** to continue.

Figure 15-19  Create a Data Transformation Map: Source or Target
6. In the Create a Data Transformation Map – Source Syntax Type window (Figure 15-20), specify the type of document that is the input to the map. In this case, we select **EDI Standard**.

**Note:** Select EDI Standard even if you selected the Source map in Figure 15-19. The option Source or Target only controls the execution order. In either case, you still want to translate EDI into XML.

Click **Next** to proceed.

![Create a Data Transformation Map - Source Syntax Type](image)

*Figure 15-20  Create a Data Transformation Map: Source Syntax Type*
7. In the Create a Data Transformation Map – Source Dictionary window (Figure 15-21), identify the dictionary that holds the definition of the source document. Select the name of the EDI dictionary, which is X12V4R1. Click Next.

![Create a Data Transformation Map - Source Dictionary](image)

*Figure 15-21 Create a Data Transformation Map: Source Dictionary*
8. In our setup, the EDI dictionary only contained a definition for the 850 transaction, which is the one we need. You might have imported the full set of transactions of the X12V4R1 standard. In the Create a Data Transformation Map – Source EDI Standard Transaction window (Figure 15-22), select 850 and Purchase Order. Then click Next.

![Figure 15-22 Create a Data Transformation Map: Source Transaction](image)

9. A similar process follows for the target document. Select the syntax type of the target document, which is XML in this scenario (Figure 15-23). Click Next.

![Figure 15-23 Create a Data Transformation Map: Target Syntax Type](image)
10. In the Create a Data Transformation Map – Target Dictionary window (Figure 15-24), select the dictionary that holds the document definition. Click Next.

![Create a Data Transformation Map - Target Dictionary](image1)

**Figure 15-24  Create a Data Transformation Map: Target Dictionary**

11. In the Create a Data Transformation Map – Target XML Schema or DTD window (Figure 15-25), select the DTD that was imported previously and click Next.

![Create a Data Transformation Map - Target XML Schema or DTD](image2)

**Figure 15-25  Create a Data Transformation Map: Target XML DTD**
12. You see a summary (Figure 15-26), which gives you the opportunity to review the selections made during the previous steps. Click **Finish** to end the map definition wizard.

![Figure 15-26 Create a Data Transformation Map: Confirmation](image)

The map is now be created and loaded into the client interface. You should now see a window similar to Figure 15-27. The top left and top right pane show the structure of input and output documents. The bottom left pane contains the actual map, from a target point of view. The contents of this pane are controlled by the setting that was done in Figure 15-19 on page 446.

Since we chose Target, you see the target XML document. By looking at that tree, you can easily see what elements still need to be mapped. By adding map statements, blue check marks appear to help remind you what has been done.

Creating map commands is done by dragging elements from the top left pane (the source) into the top right pane (the target). Elements, in this context, are the fields at the lowest level in the XML or EDI tree. Expand the trees before dragging any elements.

The bottom right pane contains a number of variables that are either created by you to assist in the mapping or that are part of the system. Two of those system variables are listed: DIOutType and DIOutFile. You can use these variables to decide in the map where the transformed document should be delivered.
Before you build the actual map commands, populate the two special variables. As explained before, WebSphere Data Interchange can read source documents and write target documents to files and queues. However, in our environment, we used queues to facilitate the transfer of documents between the WebSphere Data Interchange server, the WebSphere BI Connect server, and any back-end applications.

Therefore, the variable DIOutType needs the value MQ. The variable DIOutFile should contain the name of an MQ profile defined in the WebSphere Data Interchange database. That MQ profile should point to the queue XML_OUT that was created during the setup of WebSphere Data Interchange. Such an MQ queue profile already exists as part of the standard setup of WebSphere Data Interchange. It is called XML_OUT.

To set the value of those variables, perform the following steps.

1. In the bottom left pane of the Data Transformation Map window (Figure 15-27), right-click the map name EDI-XML-850 and select Insert → Command Group.
2. The Command Group Editor window (Figure 15-28) opens. Provide a brief description, for example, Routing information, and click **Insert**. The description will be used in the GUI.

![Command Group Editor](image)

*Figure 15-28  Command Group Editor*

3. A new label, **Routing information**, is now available in the bottom left pane of the Data Transformation Map window. Right-click it and select **Insert Within → Command → Assignment**.

4. The Mapping Command Editor window (Figure 15-29) opens. It contains a sample assignment command to help you build the command. Figure 15-29 shows the command to set the value of the DIOutFile variable. Click **OK**.

![Mapping Command Editor](image)

*Figure 15-29  Command editor*
5. Insert another assignment command to set the value of the variable DIOutType. The mapping now looks like the example in Figure 15-30.

```plaintext
EDI-XML-850
  Routing information
    DIOutType = "MQ"
    DIOutFile = "XML_OUT"
  purchaseOrder [(fileHeader,orderHeader,lineDetail*,lineText*,fileTrailer)]
    fileHeader [(date,time)]
    orderHeader [(orderNumber,orderFlag,orderDate,contractNumber,shipToAdr,]
    lineDetail [(lineNumber,partNumber,itemDesc,quantity,uom,unitPrice,] ]
    lineText [(orderNumber,lineNumber_text)]
    fileTrailer [(totalDollars,totalQuantity)]
```

*Figure 15-30  Mapping commands with routing information*
6. Build the actual mapping commands.
   a. Expand the tree structures in the top panes of the Data Transformation Map window (Figure 15-27 on page 452), so that both the target element and the source element are visible.
   b. Drag the source element and drop it onto the target element. This results in a new command that is inserted in the bottom left pane (Figure 15-31).

Figure 15-31 Building a mapping command
In addition to moving element values from source to target, you can also set the value of a target element. As an example, populate the XML element time by invoking the built-in time() function of WebSphere Data Interchange.

1. Right-click the element **time** in the bottom left pane and select **Insert Within → Command → Assignment**.

2. The Mapping Command Editor window opens again, showing a sample command. To limit typing, you can drag the target element from the top right window and drop it in the command editor. WebSphere Data Interchange then generates the correct access path for you. Then add the invocation of the function time to the command, as shown in Figure 15-32. Click **OK** to return to the mapping editor.

![Mapping Command Editor](image)

*Figure 15-32 Invoking a built-in function*
By using the drag-and-drop technique, you can now populate other fields in the target document. Figure 15-33 shows the mapping for the XML elements fileHeader and fileTrailer.

Figure 15-33   Partially complete mapping
In the same way, you can populate the fields from the element orderHeader. Figure 15-34 shows the map statements. All values are copied from the source document.

![Diagram of orderHeader elements]

**Figure 15-34  Mappings for the element orderHeader**

The element lineDetail is a bit more complex. Here, you want to copy a list of line items without knowing how many items there are. WebSphere Data Interchange contains a specific command for that purpose. The transformation engine repeats the same block of commands for each occurrence of a repeating element in the source document. To add this command, follow these steps:

1. Right-click the **lineDetail** element and select **For Each**.

2. Mapping Command Editor window opens. Drag the source element to the editor, which is the PO1 loop shown in Figure 15-35. Click OK to return to the mapping editor.
Figure 15-35  Inserting a ForEach command

The element lineDetail contains a child called lineNumber. The source document does not contain such an element. The PO1 segment has an optional field Assigned Identification that can be used as a line number. However, our trading partner does not provide such a number. It is easy to set up a counter in WebSphere Data Interchange itself and set the XML element to the right value.

3. Define a local variable.
   a. Right-click Local Variables in the pane (bottom right) and select New.
   b. The Create New Local Variable window (Figure 15-36) opens. Provide a name for the variable, for example, lineNumber. Complete the Data Type and the Initial Value fields. Click OK.

Figure 15-36  Creating a local variable
4. Return to the target element \texttt{lineNumber} and add an assignment command. Set the element to: \texttt{NumFormat(lineNumber + 1, 0)}. All other elements within the element \texttt{lineDetail} are populated by copying values from the source document, as shown in Figure 15-37.

![Figure 15-37 Mappings for the lineDetail element](image)

The map is now complete. Save it, and select \textbf{Actions} \textrightarrow \textbf{Compile} to compile it.

### 15.3.4 Trading partner setup

Before we can set up the usage or rule for a data map, we need information about the trading partners and envelopes in the database. To review or change trading partner information, we click the \textbf{Trading Partner} button in the toolbar.

WebSphere Data Interchange ships with a default Trading Partner profile called \texttt{ANY}. You can use this profile when a map is applicable for all your trading partners. In some situations, you may need a different map for the same EDI transaction for different partners. A trading partner profile document can hold information about how to interact with this company.

On the General tab, you can specify information about Interchange ID or other Network information. The Company Info and Contacts tabs contain descriptive
information about this company. The EDI Options and WDI Proc Options tabs contain values that control the EDI document structure and the processing of the documents. For example, you can control the segment ID separator and the segment delimiter.

For the scenario in this redbook, create two new trading partner profiles: one for Company A and one for Company E. The Interchange IDs are COMPANYA and COMPANEY respectively (Figure 15-38).

![Figure 15-38 Creating a new trading partner profile](image-url)
Figure 15-39 lists the profiles in our database, including the two standard profiles ANY and WMQI.

![Figure 15-39 List of trading partner profiles](image)

### 15.3.5 Rule or usage for the transformation map

A rule or usage for a map is required for WebSphere Data Interchange so that it knows when to use a particular map.

1. While the map is opened in the mapping editor or while the map is selected in the list of maps, select **Actions → Rules/Usages**.

2. You see an empty list of rules. Select **File → New** to create a new rule.

3. In the new document (Figure 15-40), select the option to associate the rule with trading partners and select trading partner profiles for Company A and Company E. Under Properties, select **Active**. For Usage Indicator, select **Production**.

**Note:** You can associate several rules with a single map. Also, you are not required to create a rule for every possible combination of trading partners. Several other options exist to control the execution of a map and to avoid the possibility that you may need to create a huge amount of rules to cover all combinations. Refer to the product documentation for additional information.
15.3.6 Supporting definitions

You must define many other objects within WebSphere Data Interchange to support the execution of the data transformation map. For example, in the mapping commands, we referred to an MQ profile. Most of these definitions for our scenario are part of the default definitions that are provided by WebSphere Data Interchange. However, in general, you may need to add other definitions or change the defaults. The following sections review the additional definitions.

Network profile
This is the only new profile that you need to create.

1. Click the Setup button in the tool bar and select the Network tab.
2. Create a new network profile EDI_XML to handle the network along which MQ messages with an RFH header are received and sent.

Table 15-1 lists the important attributes of this network profile and the corresponding parameters. The communication routine is VANIMQ, which is used for all types of MQ interaction by WebSphere Data Interchange. The network program is EDIRFH2, which must be used when the MQ message contains an RFH header. This is the case for JMS messages generated by WebSphere BI Connect. The network parameters are RECEIVEMQ and
SENDMQ. These parameters specify which file (or mailbox profile) to use for reading and writing the messages.

**Table 15-1  Details of the network profile EDI_XML**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network ID</td>
<td>EDI_XML</td>
</tr>
<tr>
<td>Communication Routine</td>
<td>VANIMQ</td>
</tr>
<tr>
<td>Network Program</td>
<td>EDIRFH2</td>
</tr>
<tr>
<td>Network Parameters</td>
<td>RECEIVEMQ=EDI_IN, SENDMQ=XML_OUT</td>
</tr>
</tbody>
</table>

**Mailbox profiles**

Two profiles were referenced in the network profile. Both of them should already exist in the WebSphere Data Interchange database.

1. Select the **Mailboxes** tab in the setup window and review the existing definitions.

2. Update the Network ID parameter, which needs to refer to the network profile created earlier. Table 15-2 and Table 15-3 list the important attributes and their values. You can clear all other fields of any values.

**Table 15-2  Details of the mailbox profile EDI_IN**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox ID</td>
<td>EDI_IN</td>
</tr>
<tr>
<td>Network ID</td>
<td>EDI_XML</td>
</tr>
<tr>
<td>Receive File</td>
<td>EDI_IN</td>
</tr>
</tbody>
</table>

**Table 15-3  Details of the mailbox profile XML_IN**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox ID</td>
<td>XML_OUT</td>
</tr>
<tr>
<td>Network ID</td>
<td>EDI_XML</td>
</tr>
<tr>
<td>Receive File</td>
<td>XML_OUT</td>
</tr>
</tbody>
</table>
**MQSeries queue profiles**

The MQSeries Queues tab in the setup window of WebSphere Data Interchange lists the profiles for queues that WebSphere Data Interchange can use. A queue profile EDI_IN and XML_OUT exists in the default setup. You only need to update the attribute Queue Manager Name. In our scenario, that attribute needs the value wdi.queue.manager.

**Service profile**

The service profile that is called always matches the name of the receive file value specified in the inbound mailbox profile. WebSphere Data Interchange assigns this logical name to the message received by the mailbox ID assigned to the WebSphere MQ queue. The service profile called is EDI_IN. In this service profile, you should have the PERFORM command:

```
PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E)
```

This command is sufficient to call the data transformation map. The map that is called defines the output file name as XML_OUT. For this reason, enter a value for XML_OUT in the Output Files tab of the Service Profile. The value should refer to a temporary file that WebSphere Data Interchange can use.

During testing, add the TRACELEVEL(A2) option at the end of the command.

### 15.3.7 Putting it all together

The following sequence of events defines how each component interacts. We have already discussed the WebSphere BI Connect workings in detail. These events begin after the receipt of an AS2 message by WebSphere BI Connect and the message has just been placed on a WebSphere MQ queue and sent from the queue manager, partner_e.bcg.queue.manager, to the queue manager, wdi.queue.manager.

1. An EDI transaction is placed on a WebSphere MQ queue.
2. The WebSphere Data Interchange adapter has already been configured to listen to this queue and triggers the WebSphere Data Interchange server as a new message is received. The adapter discovers that a new message is on the queue and issues the command:

```
PERFORM RECEIVE AND PROCESS WHERE REQID(QUEUE_NAME) FILEID(QUEUE_NAME) CLEARFILE(Y)
```

Essentially, this command receives the EDI transaction as message from the WebSphere MQ queue and assigns this message a logical file name which is the same as the queue name. The REQID, which relates to the mailbox profile configured in WebSphere Data Interchange, is also assigned the name of the queue name.
3. The mailbox profile is called. Here, WebSphere Data Interchange can assign a new logical name to the message that has just been received. The message is assigned the name as defined in the Receive File parameter. The mailbox profile also provides details of the network ID to be used.

4. WebSphere Data Interchange then examines the network ID and determines the queue name to receive from. This is identified by the SENDMQ Network Parameter. The value here relates to the MQSeries Queues profile ID.

5. WebSphere Data Interchange examines the appropriate MQSeries queue profile and the message is read from the queue.

6. The WebSphere Data Interchange server calls the service profile that matches the logical name of the message being processed.

   The PERFORM command contained in the Service Profile determines the next actions. In our example, the PERFORM command looks like this:

   `PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E)`

   This tells the WebSphere Data Interchange server to translate the message received using an appropriate map. The INFILE parameter EDI_IN again refers to the logical file name of the message being processed. SYNTAX(E) tells the WebSphere Data Interchange server that the syntax of the message to be translated is EDI.

7. The WebSphere Data Interchange server then decides which map to use in processing the message, by determining an active rule. The rule in our case defines the trading partners involved in the transaction, including the fact that the rule is active and is used for EDI messages flagged as being production type messages.

   The rule is determined based on the factors mentioned earlier. With the data WebSphere Data Interchange has already identified about the message, a best match is determined, and a map is called.

8. The map defines the relationship between one data type and another and data transformation is performed.

9. The map also determines where the output should go. The output file is defined as XML_OUT and the output type is defined as being MQ.

10. The WebSphere Data Interchange server looks for the mailbox ID with the name of XML_OUT and again determine the network ID to be used.

11. The SENDMQ parameter of the network profile determines the MQSeries Queues profile to be used.

12. This MQSeries queue profile is examined, and the details of the queue on which the message should be placed is determined. In our case, the WebSphere MQ queue name is XML_OUT. Therefore, the result of translation is placed on XML_OUT.
15.3.8 Validating the map and the supporting setup

The infrastructure now spans four machines. To assist in testing the flow from WebSphere BI Connect Advanced of Company A, to WebSphere BI Connect Enterprise of Company E, to the queue manager partner_e.bcq.queue.manager, to the queue manager wdi.queue.manager, and to the translation map in WebSphere Data Interchange, consider stopping an intermediate component. For example, you can stop the trigger monitor on the queue manager wdi.queue.manager before you send a document.

Example 15-5 shows a sample EDI message that is sent by Company A. Copy this file into the edi_out folder on the server of Company A. Then it should arrive in the queue EDI_IN on queue manager wdi.queue.manager. This assumes that all components are running:

- WebSphere BI Connect of Company A
- Queue manager partner_e.bcq.queue.manager and associated components such as the listener and the channel initiator
- WebSphere BI Connect of Company E
- Queue manager wdi.queue.manager and associated components such as the listener

This part of the infrastructure was validated in 15.2.5, “Validating the JMS and MQ configuration” on page 435.

Example 15-5 Sample EDI message sent by Company A

```plaintext
ISA* * * * * *companya * *companye *040917*1505* *
*000000005*0*P*:!
GS*P0* *20040917*1505*5* *004010!
ST*850*0005!
BEG*00*SA*P345322**20030410!
DTM*002*20030410!
P01*001*128*EA*98.62!
P01*001*100*EA*69.62!
SE*6*0005!
GE*1*5!
IEA*1*000000005!
```
After you verify that the message is stored in the queue EDI_IN, you can restart the trigger monitor using the MQ Services application. The trigger monitor is immediately notified about the message in the queue EDI_IN and starts the WDIAdapter program. When the transformation has finished, the XML document should be available in the queue XML_OUT. Figure 15-41 shows the transformed XML document in RFHUTIL.

![XML document created by WebSphere Data Interchange](image)

**Figure 15-41  XML document created by WebSphere Data Interchange**

### 15.4 Handling the outbound document flow

This section explains how WebSphere Data Interchange handles the outbound flow. XML documents are created by a back-end application. These XML documents contain purchase orders for either Company A or Company X. Company A wants to receive the purchase orders in an EDI format, while Company X is expecting an XML document. You can implement some routing logic in the back-end application. The XML documents for Company X can be delivered by the back-end application in the queue that is read by WebSphere BI Connect, and essentially, WebSphere Data...
Interchange is bypassed. The back-end application delivers XML documents for Company A in the queue that is serviced by WebSphere Data Interchange. This enables WebSphere Data Interchange to perform the transformation and send it to WebSphere BI Connect for transmission.

However, the responsibility to route documents is really a middleware task. Business applications should not be concerned with these kinds of routing decisions. This section demonstrates how you can use WebSphere Data Interchange so that this type of routing is handled. The map that you build contains some conditional logic. If the destination is Company X, it calls a second map to perform an XML-to-XML transformation. If the destination is Company A, it calls another map to perform an XML-to-EDI transformation:

- **XML-ROUTER**: This map identifies the participant ID of the receiver. Based on the value received, the WebSphere Data Interchange server calls one of two other maps.
- **XML-XML-850**: This map is used for transactions sent to WebSphere BI Connect Express of Company X. It takes XML as input and generates XML output. The map is essentially a one-to-one map where the input is mirrored in the output.
- **XML-EDI-850**: This map is called for transactions destined for WebSphere BI Connect Advanced of Company A. XML is taken as input and translated to an ANSI X12 850 transaction.

### 15.4.1 Detailed review of the map XML-XML-850

This is a simple XML-to-XML data transformation map. The data format used for both the source and target are identical. Essentially, each element in the XML format is mapped onto itself, creating a mirror image of the original document. The output file generated by this map, and the type of file are both identified in the Routing Information command group. The DIOutFile special variable identifies the output file as having the logical file name WDI_OUT. This logical file name is later associated with a physical file name, under the Output Files tab of the service profile being invoked.

When creating this map, use the following values:

- Target-based
- Source syntax type: XML
- Source dictionary: ITSO
- Source document definition: PurchaseOrder
- Target syntax type: XML
- Target dictionary: ITSO
- Target document definition: PurchaseOrder
The lineDetail and lineText elements have a ForEach command. This ensures that all occurrences of these elements are copied, not just the first one. All other elements in the target document are copied from their equivalent element in the source document (Figure 15-42).

```
<XML-XML-850>
  <RoutingInformation>
    <DIOutType>"IN"</DIOutType>
    <DIOutFile>"WEB"</DIOutFile>
  </RoutingInformation>
  <purchaseOrder[{"FileHeader","orderHeader","lineDetail","lineText","fileTrailer"}]>
    <fileHeader[{"GateOrder"]}]
    <orderHeader[{"OrderNumber","orderFlag","orderDate","contractNumber","supplierNumber","supplierId","supplierAddress1","supplierAddress2"}]
      <forEach(PurchaseOrder[lineDetail])
        <lineDetail[{"lineNumber","partNumber","lineDesc","quantity","unitPrice","dueDate"}]
          <forEach(PurchaseOrder[lineText])
            <lineText[{"orderNumber","lineNumber","text"}]
              <fileTrailer[{"totalDollars","totalQuantity"]}
            <![CDATA[Figure 15-42  Mapping the XML document to an XML document]]>
          </forEach>
        </forEach>
      </forEach>
    </orderHeader>
  </purchaseOrder>
</XML-XML-850>
```

### 15.4.2 Detailed review of the map XML-EDI-850

This map takes XML as a source and maps to an ANSI X12 850 Purchase Order transaction. Use the following values when creating this map:

- Target-based
- Source syntax type: XML
- Source dictionary: ITSO
- Source document definition: PurchaseOrder
- Target syntax type: EDI Standard
- Target dictionary: X12V4R1
- Target EDI Standard Transaction: 850

The Routing Information command group again defines the DIOutType and DIOutFile special variables. These variables are set to the same values as for the map XML-XML-850. Thus, the EDI document and the XML document are sent to WebSphere BI Connect via the same queue.

The Envelope Properties command group is used to build the ANSI X12 envelope for the resulting EDI transaction. The SetProperty() function is used to set the Interchange Sender and Receiver IDs. Their values are copied from specific fields in the XML document. This assumes that the back-end application and WebSphere Data Interchange and WebSphere BI Connect use the same
business identifiers. This may not always be the case. Refer to 15.5, “Using a translation table” on page 479, which describes a solution for that situation.

Figure 15-43 shows the routing and envelope commands, as well as the mapping for the BEG segment in the 850 transaction.
Further down in the EDI 850 transaction, you have the DTM segment for which the mapping is shown in Figure 15-44.

![Figure 15-44](image-url)
Figure 15-45 shows the last part of the mapping commands to build an EDI document. It lists the mapping commands required to map all occurrences of the element lineDetail to the PO1 loop. Only a limited number of attributes are populated, but this is sufficient to demonstrate the routing and mapping features of WebSphere Data Interchange.

Both maps XML-XML-850 and XML-EDI-850 should produce output on the same WebSphere MQ queue. This queue is a remote queue defined on the WebSphere Data Interchange queue manager. This remote queue points to a queue on the WebSphere BI Connect Enterprise queue manager. The queue on the WebSphere BI Connect Enterprise queue manager is defined as a JMS target and is processed by WebSphere BI Connect. Depending on the type of message received, the WebSphere BI Connect Enterprise packages it appropriately and sends it to the correct participant.
15.4.3 Detailed review of the map XML-ROUTER

As the name suggests, this map is used to route the XML received by the WebSphere Data Interchange server. Based on the information in the map, it calls the XML-XML-850 or the XML-EDI-850 command via the command MapSwitch().

Use the following values when creating this map:

- Target-based
- Source syntax type: XML
- Source dictionary: ITSO
- Source document definition: PurchaseOrder
- Target syntax type: XML
- Target dictionary: ITSO
- Target document definition: PurchaseOrder

**Note:** The settings for the target document are not really relevant, since this map does not do any actual mapping. This map calls other maps based on the contents of the source document. Since we use a MapSwitch() command, the target document is never accessed or built by this map. The actual target document information is picked by the map to which we have switched.

The actual commands used to determine the map that is invoked are defined in the Routing Information command group (see Figure 15-46).

![Figure 15-46 If-then-else command in the router map](image)

1. After inserting the command group, right-click the group and select **Insert Within → Command → If** (see Figure 15-47).

2. The command editor opens. Drag the supplierId element from the source document and drop it in the command editor. Its value is compared to the string value “companyx”. If the string comparison returns 0, the values are the same.
3. Within the **If** command, add a **MapSwitch()** command. Right-click the **If** command and select **Insert Within → Command → MapSwitch**.

4. Add the name of the map, XML-XML-850, in the parameters of the **MapSwitch** command.

5. Add the **Else** command. Right-click the **If** command and select **Insert After → Command → Else**.

6. Add another **MapSwitch** command within the **Else** command. This time, switch to the map XML-EDI-850.

The condition logic is designed so that the XML document is translated to EDI except for the single company called companyx.

![Building an If command](image)

**Figure 15-47  Building an If command**

The **MapSwitch()** command passes the entire source document to the new map. **Mapping commands that are defined in the map to which the WebSphere Data Interchange server switches are executed as though this was the first map identified by the rule.**

This is different for the **MapCall()** command. The **MapCall()** command can be considered as a subroutine. The called map inherits the environment of the calling map, and when the called map is finished, the context returns to the calling map. The behavior of the **MapSwitch()** command is more like a “reset: start all over again with the new map”, instead of the map determined by the rule or usage.

No rule or usage was created for the maps XML-XML-850 and XML-EDI-850. However, you must create a rule for the map XML-ROUTER. The map rule should be associated with trading partners. The sending trading partner should be set to ANY, as should the receiving trading partner. In this way, the rule used to call this map is not tied to any particular trading partner.

In 15.3.5, “Rule or usage for the transformation map” on page 462, the rule is tied to trading partners. But this time, the commands within the map are used to make any decisions.
1. In the Data Transformation Map Rule window, click the General tab.
   a. For Sending and Receiving, choose ANY, to indicate ANY to ANY. This allows any XML file whose root tag matches that of the source document type in this map to call this data transformation map.
   b. Set the Usage Indicator to Production.
   c. Under Properties, select Active for the map (Figure 15-48).

![Figure 15-48 Data transformation rule (Part 1 of 2)]
2. Since this map rule *may* result in an EDI file, make sure that the EDI document is packaged in the correct type of envelope. Select the **Envelope Attributes** tab. See Figure 15-49.
   a. Set the envelope type to X, which corresponds to an X12 style of envelope.
   b. You can also select any specific X-type envelope profiles if you defined them in the setup window. However, for this scenario, a generic X12 envelope is sufficient.

   This setting is ignored if the target document is an XML document.

![Figure 15-49 Data transformation rule (Part 2 of 2)]

### 15.4.4 Additional supporting definitions

All messages received from our back-end application arrive at the WebSphere Data Interchange server on the WebSphere MQ queue XML_IN. This queue normally exists as part of the standard installation of WebSphere Data Interchange on the queue manager wdi.queue.manager. Similarly to the inbound flow, you must add or adjust a number of profiles in WebSphere Data Interchange to support the outbound flow:

- Network profile
- Mailbox profiles
MQSeries queue profiles
Service profile

Network profile
A new network profile is required for this scenario. Use the values listed in Table 15-4 to create a profile for XML_EDI. This time we use EDIMQSR as the network program. This means that the MQ message does not have an RFH header.

Table 15-4  Details of the network profile

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network ID</td>
<td>XML_EDI</td>
</tr>
<tr>
<td>Communication Routine</td>
<td>VANIMQ</td>
</tr>
<tr>
<td>Network Program</td>
<td>EDIMQSR</td>
</tr>
<tr>
<td>Network Parameters</td>
<td>RECEIVEMQ=XML_IN</td>
</tr>
<tr>
<td></td>
<td>SENDMQ=WDI_OUT</td>
</tr>
</tbody>
</table>

Mailbox profiles
The XML_IN mailbox profile should already exist as part of the standard installation. The XML_IN mailbox profile should have the properties listed in Table 15-5.

Table 15-5  Details of the mailbox profile XML_IN

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox ID</td>
<td>XML_IN</td>
</tr>
<tr>
<td>Network ID</td>
<td>XML_EDI</td>
</tr>
<tr>
<td>Receive File</td>
<td>XML_IN</td>
</tr>
</tbody>
</table>

Create a new mailbox profile WDI_OUT. The important attributes and values are listed in Table 15-6.

Table 15-6  Details of the mailbox profile WDI_OUT

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mailbox ID</td>
<td>WDI_OUT</td>
</tr>
<tr>
<td>Network ID</td>
<td>XML_EDI</td>
</tr>
<tr>
<td>Receive File</td>
<td>WDI_OUT</td>
</tr>
</tbody>
</table>
MQSeries queue profiles
The MQSeries queue profiles XML_IN should already exist as part of the default installation. Create a new MQSeries queue profile WDI_OUT. Table 15-7 and Table 15-8 list the values appropriate for our scenario. The profile WDI_OUT refers to the remote queue TO.EDI_OUT, which refers to the local queue EDI_OUT on the WebSphere BI Connect queue manager partner_e.bcg.queue.manager. Refer to Figure 15-4 on page 426 for details about the WebSphere MQ setup.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Profile ID</td>
<td>XML_IN</td>
</tr>
<tr>
<td>Full Queue Name</td>
<td>XML_IN</td>
</tr>
<tr>
<td>Queue Manager Name</td>
<td>wdi.queue.manager</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Profile ID</td>
<td>WDI_OUT</td>
</tr>
<tr>
<td>Full Queue Name</td>
<td>TO.EDI_OUT</td>
</tr>
<tr>
<td>Queue Manager Name</td>
<td>wdi.queue.manager</td>
</tr>
</tbody>
</table>

Service profile
The service profile XML_IN should already exist. Review the PERFORM command. It should refer to the infile XML_IN. The expected syntax is XML:

PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X)

Make sure that the file WDI_OUT is listed on the Output Files tab of the service profile. The logical name WDI_OUT is defined as the output file in each of the data transformation maps detailed earlier. Without this definition, the WebSphere Data Interchange server is unable to send the result of translation to a queue.

15.5 Using a translation table
In 15.4, “Handling the outbound document flow” on page 468, we assumed that the information about business IDs is shared and synchronized between the back-end application, WebSphere Data Interchange and WebSphere BI
Connect. In reality, you may see many situations where that is not the case. The back-end application may know business partner Company A as TP_A, for example. But the business ID that is required in AS2 is companya. In this situation, WebSphere Data Interchange can provide the required mapping between both IDs.

WebSphere Data Interchange provides a feature called a Forward Translation table. This table lists both IDs. In the mapping, you use the Translate() function to map the incoming ID to the outgoing ID.

As an example, consider the case where the back-end application sends an XML purchase order to WebSphere Data Interchange, and the element supplierId contains the value TP_A. In WebSphere BI Connect, Company A is known as companya.

1. Create the translation table in WebSphere Data Interchange. Open the mapping window and select the Forward Translation tab.

2. Select File → New to create a new table.

3. On the General tab (Figure 15-50), complete these items:
   a. Provide a name for the new table, for example, LOOKUP.
   b. For Local Value, enter TP_A.
   c. For Standards or Trading Partner Value, enter the identifier companya.
   d. Save the new table.

![Forward Translation Table](image-url)
4. Amend the XML-EDI-850 map. Open the XML-EDI-850 map.

   a. Change the SetProperty() function for the Interchange Receiver ID to use the Translate() function.

   b. Amend the mapping command to appear as shown here:

   
   ```java
   SetProperty ("IchgRcvrId", Translate ("LOOKUP", "SOURCE", \purchaseOrder\orderHeader\supplierId\supplierId.PCDATA\))
   ```

   Figure 15-51 shows the updated command group for the map XML-EDI-850.

   ![Figure 15-51 Using the translate function in a map](image)

   LOOKUP is the name of the Forward Translation table being used. The SOURCE parameter indicates that the value being passed is the source value in the translation table. The last parameter that is passed to the SetProperty() function is the value to look up. This function returns the Standards or Trading Partner Value as defined in the translation table. This value is then used when building the envelope data for the transaction. It allows the WebSphere BI Connect Enterprise determine the sender and receiver IDs.

### 15.5.1 Validating the maps and supporting definitions

To isolate the testing, consider stopping at least the Receiver component of WebSphere BI Connect of Company E. This means that the transformed EDI document stays in the EDI_OUT queue on the partner_e.bcg.queue.manager queue manager. This enables you to verify the message transformation before you try to send it by WebSphere BI Connect.

Figure 15-52 shows a sample XML message. The element supplierId has the value TP_A, which is not a known business identifier in WebSphere BI Connect. Thus, besides transforming the XML document into an EDI 850 document, WebSphere Data Interchange needs to replace TP_A with companya.
When the message shown in Figure 15-52 is written to the queue XML_IN, the trigger monitor again launches the WDIAdapter program that executes the desired transformation.

Example 15-6 shows the report that WebSphere Data Interchange generates as part of the transformation. WebSphere Data Interchange finds that the map XML-ROUTER is the best match for the document PURCHASEORDER, which is part of the dictionary ITSO. First, the map XML-ROUTER is executed. But then, the report lists the map XML-EDI-850 as the executing map, which shows that the conditional logic has been executed and that the map XML-ROUTER has switched correctly to the map XML-EDI-850.

The transformation completes successfully. The document is written to the WDI_OUT logical file, which maps to the TO.EDI_OUT queue, which is the remote queue object that points to the queue EDI_OUT on the partner_e.bcg.queue.manager queue manager.
**Example 15-6  Audit report of the transformation**


FF0588 Command: PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X) TRACELEVEL(A2)

Message: RU0003 Severity: 00
The best rule match for the document was: map name XML-ROUTER, sending TP nickname ANY, receiving TP nickname ANY, usage indicator P, document PURCHASEORDER, dictionary name ITSO, syntax xml.

Message: UT0008 Severity: 00
Map name being processed: XML-ROUTER.

Message: UT0008 Severity: 00
Map name being processed: XML-EDI-850.

FF0007 Data was written to WDI_OUT. Message control number or document id was 000000007.

FF0585 The PERFORM TRANSFORM command completed successfully.

FF0020 Completed sending network files for WDI_OUT.

Figure 15-53 shows the transformed EDI document. The sender and receiver ID are set correctly to the business identifiers that WebSphere BI Connect knows.

![Figure 15-53  EDI document after translation and before sending](image)

Chapter 15. Integration with WebSphere Data Interchange  483
Example 15-7 shows what WebSphere Data Interchange adds to the usr folder in the RFH header. Again you see that the sender ID and receiver ID are populated correctly.

Example 15-7  Contents of usr folder

```xml
<SenderId>companye</SenderId><SenderQual/>
</SenderQual><ReceiverId>companya</ReceiverId><ReceiverQual/></ReceiverQual>
```

You can now restart the WebSphere BI Connect Receiver component that will pick up the message from the queue EDI_OUT and deliver it to Company A.

### 15.6 Adding a new EDI transaction

The previous sections looked at document routing for the outbound flow so that the outbound document is in the right format for the intended partner. For the outbound flow, we did not need to consider a different target location. WebSphere BI Connect can pick up any document type from a queue and determine the format that it has and what to do with it. However, for the inbound document flow, it may be easier for back-end applications if different types of documents are delivered in different queues.

Assume the following situation. Company A sends two different EDI transactions to Company E, for example 850 and 846. Company E uses WebSphere Data Interchange to translate the incoming EDI documents in an appropriate XML format for processing by back-end applications. However, the business applications reside on different systems. Or they are separated in another way so that it is required to drop the XML version of the EDI 846 document in a different queue than the XML version of the EDI 850 document.

The solution is easy to implement. Since each document type is handled by a different map, it is sufficient to set the system variable DIOutFile to a different value, for example 846_OUT (Figure 15-54).
Figure 15-54  Mapping EDI 846 to XML

The rule for this map is similar to the rule created for the map EDI-XML-850. The other differences, compared to the 850 translation, are in the supporting definitions. We already have mailbox profile EDI_IN that is reused for this scenario. A new mailbox profile is required, for example 846_OUT. This mailbox profile refers to the network ID 846_XML and to the receive file 846_OUT. The network profile 846_XML has the same communication routine and network program as the network profile EDI_XML.

However, the network parameters are slightly different. This time, the network parameters are:

RECEIVEMQ=EDI_IN SENDMQ=846_OUT

A new MQSeries queue profile 846_OUT is required. This queue profile refers to a new WebSphere MQ queue, for example, 846_OUT, which is the input queue for the application that handles inventory inquiries.

Finally, the last change is the service profile. We have service profile EDI_IN, but need to add a new file to the tab Output Files of this service profile. Add 846_OUT to the list of output files to complete the changes to support this transformation from EDI 846 to XML.
15.7 Summary

This chapter looked at several techniques where WebSphere Data Interchange provides additional services to a B2B solution based on WebSphere BI Connect.

WebSphere Data Interchange can provide transformation services so that documents have the right format for either the back-end applications or for the target business partners. WebSphere Data Interchange can transform a given document differently based on the target partner so that each business partner receives a document in his preferred format.

WebSphere Data Interchange can also handle differences in trading partner profiles. If a partner is known under one name in the back-end application and under another name in the B2B gateway, then WebSphere Data Interchange can perform a lookup to adjust the document.

Finally, this chapter looked at document routing to different back-end systems. If a purchase order needs to be sent to a different destination than an inventory inquiry, then WebSphere Data Interchange can route the document accordingly, even if WebSphere Data Interchange retrieves the incoming 850 and 846 documents from the same location.
Extending the features of WebSphere BI Connect
Chapter 16. Introduction to the user exit framework

The goal of B2B integration is simple: to electronically exchange information securely between one or more participants in a given community. The implementation of this goal is challenging. This is due to a number of factors:

- Number of competing protocol packaging and standards
- Different security requirements between participants
- Varying number of transports
- Varying number of message requirements and formats

With the capabilities of the base product and the inclusion of user exits in version 4.2.2 of the product, WebSphere BI Connect provides the necessary capabilities to meet these challenges for RosettaNet, AS2, and custom protocol development.
16.1 User exits and document flow

In WebSphere BI Connect version 4.2.1, users and community managers had the ability to configure a set product base. With the release of version 4.2.2 of WebSphere BI Connect, users and community managers are empowered to extend the provided functionality of the product by leveraging new WebSphere BI Connect APIs and through the use of a number of well placed user exits.

User exits are breaks in the natural process flow that allow developers to insert custom requirements. User exits add an infinite amount of flexibility to the existing rich product capabilities that are delivered in the base WebSphere BI Connect product. You can develop them to extend or modify an existing capability or to create a whole new WebSphere BI Connect offering.

Before discussing the specifics of this new capability, let us review a typical WebSphere BI Connect process flow. We continue to use the scenario that has been used throughout this redbook. Let us revisit this B2B exchange between Company A to Company E and discuss the AS2 document flow that these companies have created. The exchange follows the flow as illustrated in Figure 16-1 and explained in the following list.

![Diagram of Company A to Company E process flow]

**Figure 16-1 Company A to Company E process flow**

1. Company A’s back-end application creates a EDI X12 file and places the file on a file system.

2. A file system receiver polls the file system looking for new files. When a new file is detected, the receiver retrieves the file and passes the file to the document manager for processing.
3. The required packaging and protocol information is pulled from the incoming document and its packaging.

4. Knowing who the document is from, the intended destination and the associated protocol, WebSphere BI Connect identifies the appropriate participant connection and the underlying interaction. It then calls the appropriate action as part of the Variable Workflow. In this case, a Pass Through action is set.

5. The business document is passed to the outbound fixed workflow where AS2 packaging takes place.

6. The document is passed to the WebSphere BI Connect provided HTTP sender and sent to Company E.

7. Company E receives the HTTP request via the WebSphere BI Connect provided HTTP receiver. The receiver sets the necessary transport headers so that the document manager can use it to determine the required processing. This information is stored in meta files about the incoming document and used by the document manager to process the document.

8. The business document is then passed to the Inbound Fixed Workflow where the document is unpackaged and the protocol headers are read.

9. The document is then delivered to a pass-through action going to Company E’s back-end application.

10. The outbound fixed workflow provided by WebSphere BI Connect is called.

11. A file system sender is called to create a file and place the file on the file system where Company E’s back-end application expects new inbound requests.

12. Company E’s back-end application retrieves the new file and processes the request.

### 16.2 User exits and components

User exits are strategically placed in the system along this document flow. Before discussing the specifics of user exits and WebSphere BI Connect components, we must mention a few other terms that are used often in the following sections.

Existing components can be modified with the use of **handlers**. Handlers are code modules that may be written and deployed to various components of the solution. Handlers are configured to be used in various spots in the process flow by way of **configuration points**. Configuration points are used to configure a **handler chain**.
WebSphere BI Connect calls each handler in the handler chain in the configured order until it finds a handler that is prepared to service the request.

The availability of configuration points varies by each WebSphere BI Connect component. We discuss the use of configuration points and handlers as they apply in each of the following sections.

**Note:** When using configuration points and custom handlers, a mix of user-defined and WebSphere BI Connect supplied handlers may be configured.

### 16.2.1 Receivers

Receivers facilitate the acceptance of inbound documents. Receivers provide the entry point for all inbound document flows from both partners and back-end systems. WebSphere BI Connect includes the following receivers:

- HTTP/S
- Java Message Service (JMS)
- FileSystem
- FTP Directory
- SMTP
Receivers are responsible for accepting the inbound document from a particular transport. A target is an instance of a receiver configured for a particular deployment. Since targets are instances of a given receiver, you may have $n$ targets configured from one receiver.

For example, you may configure multiple JMS targets using the JMS receiver. Each JMS target can be configured to poll for inbound documents on separate inbound queues. Each target’s configuration specifies which inbound queue the target polls. See Figure 16-3.

![Figure 16-3  Multiple targets polling data from the same source](image)

**Important:** Notice that two targets are polling the same queue. When developing custom receivers, developers must consider the possibility that multiple targets may compete for the same inbound document and manage this condition accordingly. You can also have two instances of the Receiver component of WebSphere BI Connect. The same target is then running in both instances of the Receiver component.

Developers may extend the functionality of a provided receiver by implementing and deploying a handler and setting the desired configuration point. Receivers provide three configuration points where new handlers may be configured (see Figure 16-4):

- **Preprocess:** The preprocess handler is executed after the document is received on the transport and before the document is passed on to the document manager. A common preprocess activity is to parse a batch document that contains multiple transactions and create multiple flows containing one inbound transaction per document.
- **Synch-check**: The synch-check handler returns a boolean value and is executed after the preprocess handler is finished executing. If the sender of the document expects a synchronous response, the synch-check handler should return `true`. For example, in the case of an HTTP target, if the synch-check handler returns true, WebSphere BI Connect system caches the request connection and retrieves this cached connection when the response is returned from the document process.

- **Postprocess**: A postprocess handler is executed before a synchronous response is sent back the original requestor.

**Note**: The HTTP receiver is the only receiver where user-defined handlers can be introduced in 4.2.2. Starting with fix pack 3, however, all product provided senders and receivers are enabled.

Notice a WebSphere BI Connect provided receiver can call a combination of WebSphere BI Connect provided handlers and custom handlers at the defined configuration points. The first handler can be one provided by the product or custom developed. The receiver framework iterates down the handler chain until a handler is found that can manage the request. The receiver framework is that component of WebSphere BI Connect that sits underneath all receivers.

Alternatively, a custom receiver may be developed by using the 4.2.2 provided receiver framework and associated APIs. Developers who want to create a new receiver have to implement the classes and methods provided by the framework. When deployed, the new receiver interacts with WebSphere BI Connect via the framework.
Custom handlers and WebSphere BI Connect provided handlers can be configured in any order. Thus, it is not necessarily true that WebSphere BI Connect provided handlers are always called first. This depends entirely on the actual configuration of the receiver.

Like the WebSphere BI Connect provided receivers, a custom or user-defined receiver can call a combination of user-defined and WebSphere BI Connect provided handlers.

**Note:** User-defined receivers may use WebSphere BI Connect provided handlers.

### 16.2.2 Inbound Fixed Workflow

Inbound Fixed Workflow is the first phase of document processing by the document manager. It is called “fixed” because the sequence of the processing in an Inbound Fixed Workflow is constant. The fixed sequence cannot be modified. Fixed inbound workflow covers the standard processing done to all documents coming into the document manager from a receiver. Inbound Fixed Workflow is primarily responsible for:

- Parse transport (HTTP or JMS) headers
- Parse protocol (AS2) wrappers
- Document decryption

Inbound Fixed Workflow can only be changed with the use of handlers. As mentioned previously, Inbound Fixed Workflow consists of a fixed number of steps as illustrated in Figure 16-5.
All messages that come into WebSphere BI Connect are packaged according the specification of a specific business protocol. For example, a RosettaNet document is packaged according to the RosettaNet Implementation Framework (RNIF) specification. Protocol unpackaging involves unpackaging the message so that it may be further processed. This process may include decryption, decompression, signature verification, extraction of routing information, user authentication, or business document parts extraction. WebSphere BI Connect provides handlers for RNIF, AS2, MIME, EAI, and NONE packaging. If handlers for other packaging types are necessary, they can be developed as user exits.

Protocol processing involves determining protocol specific information, which may include parsing the message to determine routing information (sender ID, destination ID), protocol information (business protocol and version, such as RosettaNet version V02.02), and document flow/process information (such as 3A4 version V02.02.) WebSphere BI Connect provides processing for XML, RosettaNet, and EDI. Processing for other protocols (for example, CSV) can be provided using a user exit plug-in.

16.2.3 Variable Workflow

Variable Workflow, as the name implies, contains steps, arranged in a sequence, which taken all together make up an action. A step is a specific unit of code that is executed for a defined purpose such as data transformation, document validation or duplicate document checking. Steps are sequenced to form Actions. WebSphere BI Connect provides steps and actions in the base product. Actions are specified in the console as part of the process of creating participant connections. In earlier configurations, we used the simplest action, the Pass-Through action.

Users may develop custom steps and custom actions. New actions are created by copying an existing action and making the desired changes. Some existing WebSphere BI Connect actions cannot be modified because they represent a process flow that is consistent and defined by a standards body, for example, actions that involve specific types of RosettaNet processing.

Table 16-1 lists the provided WebSphere BI Connect actions. This table indicates which actions can and cannot be modified.
Several possibilities are associated with the creation of a new Variable Workflow. First, developers may use the existing WebSphere BI Connect provided actions. They consist of a sequence of steps provided by the product as well. See Figure 16-6.

### Table 16-1 Actions provided in the base product

<table>
<thead>
<tr>
<th>Action name</th>
<th>Modify?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass Through</td>
<td>Yes</td>
</tr>
<tr>
<td>Community Manager Cancellation of RosettaNet Process</td>
<td>No</td>
</tr>
<tr>
<td>RosettaNet Pass Through with Process Logging</td>
<td>Yes</td>
</tr>
<tr>
<td>Bi-directional Translation of RosettaNet and RosettaNet Service Content with Validation</td>
<td>No</td>
</tr>
<tr>
<td>Bi-directional Translation of RosettaNet and XML with Validation</td>
<td>No</td>
</tr>
<tr>
<td>Bi-directional Translation of Custom XML with Duplicate Check and Validation</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom XML Pass Through with Duplicate Check and Validation</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom XML Pass Through with Duplicate Check</td>
<td>No</td>
</tr>
<tr>
<td>Bi-directional Translation of Custom XML with Validation</td>
<td>Yes</td>
</tr>
<tr>
<td>Bi-directional Translation of Custom XML</td>
<td>Yes</td>
</tr>
<tr>
<td>Custom XML Pass Through with Validation</td>
<td>Yes</td>
</tr>
<tr>
<td>Bi-directional Translation of Community Manager Custom XML to RosettaNet with Content Duplicate Check and Validation</td>
<td>No</td>
</tr>
</tbody>
</table>

**Figure 16-6** WebSphere BI Connect provided actions and steps
Second, the developer can copy a WebSphere BI Connect provided action (modification rules in Table 16-1) and create new steps in a new user-defined sequence or action. This user-defined action uses a combination of WebSphere BI Connect supplied and user-defined steps. See Figure 16-7.

Finally, a developer may copy the Pass Through action and create new steps. This essentially constitutes a new user-defined action. See Figure 16-8.

16.2.4 Outbound Fixed Workflow

Like Inbound Fixed Workflow, Outbound Fixed Workflow has a defined sequence of steps and is fixed. Outbound Fixed Workflow is the last step of document processing and is responsible for preparing the document for transmission. Common tasks include:

- Protocol wrapping (creating the necessary AS2 packaging for example)
- Document encryption and signing

Unlike Inbound Fixed Workflow, Outbound Fixed Workflow has one configurable step, which is protocol packaging. See Figure 16-9.
16.2.5 Senders

Senders have the opposite responsibility as receivers in that senders place outbound documents on the desired transport. Senders also interact with back-end systems and partners. WebSphere BI Connect provides these senders:

- HTTP/S
- SMTP
- JMS
- File System
- FTP/S

Instances of senders are called gateways. Like the relationship between receivers and targets, you can configure $n$ gateways from one sender. A gateway represents a configured sender.

WebSphere BI Connect provides a sender framework to create new senders based on custom requirements. You may also reuse a WebSphere BI Connect sender and optionally create custom handlers to meet your specific needs. Senders provide two configuration points where handlers may be introduced.
► **Preprocess:** A preprocess handler is called after the document is received from the document manager and before the document is sent on the transport. A developer may use this opportunity to perform a custom function before the document is sent on the desired transport to the participant.

► **Postprocess:** A postprocess handler is called in situations where the sender has sent the document and is expecting a response. When the response is returned, the sender framework invokes the handler chain in the configured order until the appropriate postprocess handler is executed.

These configuration points are just like the ones described for the receiver. WebSphere BI Connect calls the configured handlers in order until one can complete the requests. However there is a difference between the senders and receivers.

As we described earlier, custom receivers actually call the framework to execute the configured handlers. Sender handlers are called by the sender framework before (preprocessing) and after (postprocessing) the sender code is invoked. See Figure 16-10.

![Figure 16-10 WebSphere BI Connect sender framework with configuration points](image-url)
Business case and development methodology for implementing user exits

This chapter describes the appropriate methodology associated with user exit development. To illustrate the development cycle of user exits, first it introduces a fictitious business case for developing new WebSphere BI Connect extensions. This business case is used as a reference for the rest of Part 4, “Extending the features of WebSphere BI Connect” on page 487.
17.1 Business case

Business cases are used to drive the requirements process. We develop the business case so we may leverage as many of the user exits as possible to provide actual coverage of requirements. We describe a business case so we can further detail how we will use the WebSphere BI Connect user exits to solve the problem.

**Note:** The business case was developed to demonstrate as much user exit functionality as possible. As such the business case may occasionally require us to reinvent existing capabilities.

For simplicity sake, we reuse the existing company profiles that were developed in the previous chapters and add requirements for a new business protocol.

Company A and Company E want to exchange data using a custom package. Fortunately, both companies are using WebSphere BI Connect. To support the custom package, we require extensions to the WebSphere BI Connect product.

The use of user exits by one company does not imply that the trading partners of this company need to use WebSphere BI Connect as well. In general, you can develop a user exit to support a custom package in WebSphere BI Connect, while your trading partner uses a different and possibly more specialized B2B product that has built-in support for this package.

17.1.1 Use case

The custom package requirements were developed by an industry organization called ITSO. The package dictates that the participants:

- Use HTTP to send and receiver business documents
- Use an ITSO defined package
- Format business data using an XML schema

The package format and the XML schema used to represent business transactions are provided by the organization, ITSO. The ITSO package contains the necessary fields required for business to business exchange such as:

- To Participant Identifier
- From Participant Identifier
- Package Name
- Package Version
The actual business data (XML document) will be wrapped in this ITSO package.

Company A and Company E also want to send the documents encrypted to protect the contents from unauthorized viewing.

Figure 17-1 graphically represents the document flow for our use case.
Figure 17-2 shows a sample unencrypted document as it will be exchanged between Company A and Company E. The first line is the ITSO package.

**Note:** The payload refers to protocol name and version. However, this is not required. In general, this type of information can be obtained in many ways or from many sources.

```
ITSO,1.0,companya,companye,0
<?xml version='1.0' encoding='UTF-8'?>
<ITSO:ITSO_INVOICE xmlns:ITSO="http://itso.ibm.com/xml">
   <protocolName>ITSO_XML</protocolName>
   <protocolVersion>1.0</protocolVersion>
   <Version>1.0</Version>
   <From>companya</From>
   <To>companye</To>
   <AccountNumber>accountNumber</AccountNumber>
   <OrderNumber>orderNumber</OrderNumber>
   <Shipping>
      <ContactName>contactName</ContactName>
      <ContactPhone>contactPhone</ContactPhone>
      <ShippingAddress>shippingAddress</ShippingAddress>
   </Shipping>
   <Items numItems="2">
      <Item>
         <Number>item1.number</Number>
         <Description>item1.description</Description>
         <Quantity>item1.quantity</Quantity>
         <UnitPrice>item1.unitPrice</UnitPrice>
         <SubTotal>item1.subTotal</SubTotal>
      </Item>
      <Item>
         <Number>item2.number</Number>
         <Description>item2.description</Description>
         <Quantity>item2.quantity</Quantity>
         <UnitPrice>item2.unitPrice</UnitPrice>
         <SubTotal>item2.subTotal</SubTotal>
      </Item>
   </Items>
   <OrderTotal>orderTotal</OrderTotal>
</ITSO:ITSO_INVOICE
```

*Figure 17-2  Sample XML document with ITSO packaging*
Company A’s application generates a business document that needs to be sent to Company E and place the file on the file system with a predefined extension. Company A needs to extend WebSphere BI Connect to pick up the files from the file system. Additionally, Company A’s back-end application only supports creating comma separated values in files. WebSphere BI Connect requires some extensions to create the correct ITSO protocol compliant business document in XML. This XML business document must be wrapped in the ITSO package and encrypted before it is sent to Company E over HTTP.

Figure 17-3 shows a sample document as it is received by WebSphere BI Connect from the back-end application of Company A.

```
COMPA_CSV,1.0,ITSO_INVOICE,1.0,companya,companye,accountNumber,orderNumber,contactName,contactPhone,shippingAddress,2,item1.number,item1.description,item1.quantity,item1.unitPrice,item1.subTotal,item2.number,item2.description,item2.quantity,item2.unitPrice,item2.subTotal,orderTotal
```

*Figure 17-3  Sample comma separated file as delivered to WebSphere BI Connect*

Company E receives the document, decrypts the contents, removes the ITSO packaging, and forwards the request to Company E’s application. Company E’s application accepts XML documents but requires that the XML document is created on a file system with a predefined file extension.

### 17.1.2 Summary

The following summarizes some key points of the use case.

- **Data encryption**: Company A and Company E exchange encrypted data. Company A is required to encrypt data and Company E is required to decrypt data as the information is sent from Company A to Company E.

- **Custom package**: The two companies use a new custom package. This custom package wraps data and represents different message parts using comma separated values.

- **Data transformation**: Company A is required to transform the comma separated document produced by the Company A application and create an XML document that is compliant with the ITSO provided schemas.

- **Interaction with applications**: Company A and Company E need a way to interact with their respective applications via a file system.
17.2 User exit development methodology

As with all aspects of custom development, the success of any given project depends on a clear understanding of the development methodology. There are many different methodologies that are promoted within the computing industry. All essentially contain the same fundamental elements.

17.2.1 Requirement gathering

Before you can develop a solution, you must know the requirements or business objective. The more fine grained the requirements are, the less likely there will be a misrepresentation of function.

Requirements for business protocols can come from a variety of sources including business analysts and industry standards bodies, such as RosettaNet or ITSO for our use case. For our development, the requirements are derived from the business case described in 17.1, “Business case” on page 502. It is essential to the success of the project that you document the requirements and clearly articulate them to the development team. When gathering business requirements, it is common to ask the following questions:

- Is there an existing industry standard for the business protocol?
  Industry standards can provide clarity and structure to the B2B exchange. In our case, ITSO created the requirements for document packaging and the business data.
- What transports are required and does the chosen transport impose any specific processing requirements?
- What is the average volume of requests?
- What is the average transaction size?
- How is the data formatted?
- Is specific business protocol packaging required?
- How are the message sender and message receiver identified within the given message structure? For example, consider a Data Universal Numbering System (DUNS) number.
- Is data transformation required?
- Is there a possibility that duplicate messages may be received? If so, what is the appropriate course of action if a duplicate message is detected?
- Does the message require validation against a known format (XML schema validation)?
Are acknowledgements required? How are acknowledgments communicated?

Is security required? If so, how are security credentials exchanged?

Is non-repudiation required? What kind of monitoring and audit capabilities are required?

Is the exchange bi-directional?

Is the exchange synchronous? What is the appropriate course of action in case of message receiver unavailability?

Is internationalization a factor and what kind of requirements are imposed?

These questions represent a few of the areas that you must explore when developing a new B2B solution. This is not an exhaustive list but you can use it as a starting point when developing your own custom solution. Remember, the more information that you know at the beginning of the development process, the better your outcome will be. Record all requirements in a mutually agreed upon format to avoid miscommunication, misrepresentation, and scope creep.

Relating our requirements to our business case, the following points are true.

A custom ITSO package is required. Company A has to wrap the business data in the ITSO package prior to sending, and Company E has to unpackage the document to determine the source and the destination.

Security is required and the extension has to support data encryption and decryption.

Data transformation is required to transform the comma separated values used in Company A’s application to the XML structure required for the ITSO compliant exchange between Company A and Company E.

The transmissions between the partners will use existing HTTP components in WebSphere BI Connect.

The transmissions will be sent asynchronously and will not require an acknowledgement from the message recipient.

The exchange will be uni-directional. All data flow for this demonstration will flow from Company A to Company E.

For the purpose of this business case and in the interest of time, internationalization is not a focus.
17.2.2 Design

Would you build your house without a clear plan? Would you develop a new automobile without approved designs and schematics? Hopefully you answered no. Then why develop software without a clear, comprehensive development plan? Lack of time?

Statistics show that construction without requirements and design is the number one reason that software development projects fail. Besides, in many cases, cars and houses can be much cheaper than IT budgets. The point is that methodology describes a course of action that is to be followed in sequence. Sure there are some activities that can be started in parallel. Choose them wisely or suffer the consequences.

A sound design is fundamental to success. A documented design allows IT departments to share plans, divide work, and identify potential incompatibilities. Some software endeavours can take years to finish. Planning becomes an essential requirement for success. Across the span of a year, software prerequisites can change, people can change and budgets can change. Sometimes change is good, and sometimes it is bad. In any case, a comprehensive and well articulated design provides the opportunity to re-evaluate in the face of new obstacles or opportunities.

When developing WebSphere BI Connect user exits, it is important to design each piece of the solution with the overall objective in mind. Sometimes the objective is specialized, and sometimes the objective for the components is geared toward reuse across multiple business protocols. In either case, a comprehensive design can be a life saver and at worse a reason as to why the development team is pursuing a particular course of action.

It is also important to have a firm understanding of user exits and the potential and possibilities those exits present. The exits are strategically placed so that the appropriate activity can be introduced in the correct place. The design must use this flexibility and ensure that the new extensions follow best practices. For example, it is not practical to perform data transformation in the receiver. It is not practical to encrypt data in an inbound workflow step.

At this stage, we can make general assumptions about our design from the knowledge gained in Chapter 16, “Introduction to the user exit framework” on page 489, and the requirements outlined in the previous section.

Table 17-1 outlines the requirements for developing user exits.
<table>
<thead>
<tr>
<th>Requirement(s)</th>
<th>Extension required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security is required and the extensions must support document encryption and</td>
<td>Data encryption and decryption are best executed in the Fixed Workflow steps. Decryption is required when the document is received on inbound. Decryption is required before the document can be unpackaged so this is best executed in a custom protocol unpackaging step.</td>
</tr>
<tr>
<td>decryption.</td>
<td></td>
</tr>
<tr>
<td>A custom ITSO package is required. The extensions must package or unpack the</td>
<td>After the message is decrypted, the actual business document must be parsed so that business protocol headers can be determined. This requirement is met by creating a custom Protocol Processing handler.</td>
</tr>
<tr>
<td>document as appropriate.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Conversely, encryption is required before the document can be sent and after the business data is packaged in the custom business protocol wrapper. Encryption is required on the “outbound” so we create a custom outbound fixed workflow handler and introduce this handler in the protocol packaging step. These custom handlers provide the necessary functions to meet our requirements for security and custom packaging.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Data transformation is required to transform the comma separated values used</td>
<td>Data transformation, data validation and other common activities are executed in the variable workflow steps. To meet our requirements, Company A has to design a new action that calls a new custom data transformation step.</td>
</tr>
<tr>
<td>by Company A’s application to the XML business document required by ITSO.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To meet this requirement and demonstrate user exit functionality, we design a custom receiver and sender to interact with our file system. We choose a file system as the transport because file system transport does not require you to install and deploy any extraneous prerequisites such as Java Message Service (JMS) or FTP. Our custom receiver and sender are developed using their respective frameworks and application programming interface (APIs).</td>
</tr>
<tr>
<td>Company A and Company E must interact with their applications using files with</td>
<td></td>
</tr>
<tr>
<td>a fixed extension.</td>
<td></td>
</tr>
</tbody>
</table>
Figure 17-4 shows our use case in a little more detail. Note the explanation that follows for this process.

1. Company A’s application creates a comma separated file and places the file on a file system. It has a fixed extension. The comma separated file contains the necessary information to determine that Company E is the intended destination.

2. A custom receiver polls the file system looking for new files. When a new file is detected, the custom receiver retrieves the file and passes the file to the document manager for processing.

3. The comma separated file is parsed, and the values for protocol, document type and business identifiers are acquired.

4. Knowing the intended destination and the associated protocol of the inbound document, WebSphere BI Connect can look up the participant connection. The appropriate action is associated with that participant connection, so the comma separated file can be converted to the required XML format.

5. After the data transformation is complete, the business document is passed to the outbound fixed workflow where ITSO packaging and document encryption takes place.

6. The document is passed to the WebSphere BI Connect provided HTTP sender and sent to Company E.

7. Company E receives the HTTP request via the WebSphere BI Connect provided HTTP receiver, which sets the necessary transport headers. The document is still encrypted at this point.
8. The business document is passed to our custom Inbound Fixed Workflow, where package headers are read and the document is then decrypted.

9. The document is delivered to a pass-through action heading for Company E’s application. Company E’s application accepts the XML formatted data so data conversion is not required in this case.

10. WebSphere BI Connect’s provided outbound fixed workflow is called.

11. A custom sender is called to create an XML file and place the file on the file system where Company E’s application expects new inbound request.

12. Company E’s application retrieves the new file and process the request.

For the most part, developers from Company A and Company E are required to implement complimentary components. For example, Company A will encrypt the documents that are sent to Company E. Company E must decrypt the document. Company A has to package the business document in the ITSO required format. Company E has to unpackage the ITSO required format. An exception is that Company A is required to convert the delimited data from Company A’s application to the required XML format.

17.2.3 Implementation

Implementation is the means to our goal. Commonly called construction, it is where we get our hands dirty. During the implementation phase, we write code, try new things and hopefully validate our design in the process.

The development life cycle is not strictly forward only. Many times we discover things in construction that require us to modify our design and redistribute. That is OK. Since we have a well articulated design, we can disseminate these changes so that other members of the development team can review them to assess impact. This allows independent parts, especially globally distributed teams, to work in harmony.

Change is inevitable. Failing due to change is avoidable. We talk more about our implementation later.

User exit development

Our development team uses WebSphere Studio Application Developer to create the WebSphere BI Connect extensions. In a common customer development scenario, Company A and Company E would likely have separate development teams and code the extension specific to their own unique perspective. For example, Company E does not require CSV to XML transformation.

For simplicity, our development team created all extensions needed to support Company A and Company E’s business requirements in one project. We will
export and distribute the resulting Java Archive (JAR) file from our project to both Company A and Company E’s WebSphere BI Connect instances.

We added these JAR files to the project build path as shown in Figure 17-5:

- **bcgskdk.jar**: JAR file distributed with WebSphere BI Connect
- **JAXMe jars**: Used to manipulate XML documents
- **JUnit.jar**: JAR file used to develop and run test cases
- **log4j-1.2.8.jar**: JAR file used to write log records into the logging subsystem that is used by WebSphere BI Connect
- **xercesImpl.jar** and **xml-apis.jar**: Two more JAR files used for manipulating XML documents

![Java Build Path](image)

**Figure 17-5Java build path and added libraries**

**Important:** We placed these files in the C:\java\lib directory. This is important because we will use ANT to generate code for our use with the JAXMe libraries. The JAXMe libraries are used for manipulating XML documents. You can download them from the Web at:

http://ws.apache.org/jaxme/
We use CVS when developing as a group and for source control. Developers check code into the source control system at regular intervals to ensure that work is saved to a common place.

17.2.4 Testing methodology

Preparation for testing begins with requirements and design. Functionally, we know what needs to be tested due to the requirements. Technically, we must ensure that the solution is robust and built with error handling and fault tolerance in mind.

In B2B integration, errors can have a devastating impact to the bottom line.

- Orders may not be sent.
- Outages can cost a company a tremendous amount of revenue in lost sales, opportunity, or both.
- Payments may not be received.

Testing is vitally important to any solution. There are multiple phases of testing.

- **Unit tests**, which can come in various forms, are executed by the developer.
  - **White box testing**: A developer may use specialized tools and classes, such as JUnit, to write and execute white box testing. JUnit is especially useful in that it provides the ability to quickly regression test existing code in the face of modifications. When developing JUnit tests, developers are encouraged to test each function and return an assertion. Assertion returns True or False. If the value of an assertion changes after a code modification, it is a clear sign that the code should be reviewed.
  - **Black box testing**: This is the traditional approach to testing. Developers try to create black box unit tests without the benefit of the detailed knowledge of code.

- **Integration tests** are end-to-end tests that demonstrate the complete flow. Integration test scenarios are usually developed with the high level business requirements in mind.

17.2.5 Error handling, problem determination in WebSphere BI Connect

Receivers, workflow components and senders use common methods for problem determination and error handling.
Events and exceptions
Events are different than exceptions. Events encompass exceptions and are generally raised when any activity of interest has occurred. The following are examples of when to raise events:

- Entering or exiting a method or routine
- A non-fatal warning or informational messages
- An exception

Exceptions are generated when an error in processing has occurred. When the user exit component developer detects an exception, the developer should act appropriately given the type of exception and log an event if possible.

Events
An important consideration during design and planning for test activities is to provide the appropriate level of debug and event messages in the code. WebSphere BI Connect uses the classes Events and EventInfo to raise three types of messages:

- **Informational events**: These events are used to raise awareness to the internal working of the components without failure. These are similar to debug messages raised at regular intervals. It is a common practice to raise informational events when the flow control of the processing enters and exits a particular function.

- **Warning events**: These events are raised based on processing logic. Warning events can be used for a variety of purposes and are specific to the component and the nature of the processing. For example, a developer may want to raise a warning message if a JMS receiver fails to close a connection to a queue.

- **Error events**: These events are hard errors that must be raised if the component is unable to continue processing the document. The nature and number of these errors depend on the components processing logic and responsibilities. For example, an error event should be raised by a JMS receiver if the receiver is unable to determine the reply-to queue manager of a synchronous request.

Events are created using the EventInfo class. The EventInfo class can be instantiated as shown in Figure 17-6.

When the event is created, there are two different ways to raise the event depending on the nature of the activity and the user exit type. Developers may use the bcgUtil class to raise a general event. In all user exits where a BusinessDocumentInterface object is available, a developer may associate an event with a business document by using the BusinessDocumentInterface.addEvent() method, as shown in Figure 17-6.
For Receivers, the ReceiverDocumentInterface.addEvent() method is used to associate the event to a particular inbound document. For senders, the SendResult.addEvent() method is used to associate an event with sender processing.

```java
ei = new EventInfo("BCG240612", busDoc, params);
eiArr = new EventInfo[1];
eiArr[0] = ei;
busDoc.addEvents(eiArr);
```

**Figure 17-6  Creating an instance of EventInfo**

As you will see, you can view events using the WebSphere BI Connect Console.

**Standards for event creation**

WebSphere BI Connect provides events and the associated event codes for each exit type.

The following events were created for receiver development:

- **Informational events**
  - BCG103207 - Receiver Entrance
  - BCG103208 - Receiver Exit

- **Warning events**
  - BCG103204 - Target Processing Warning

- **Error events**
  - BCG103203 - Target Processing Error
  - BCG103205 - Target Error

The following events were created for workflow development:

- **Informational events**
  - BCG240603 - Packaging Business Process Entrance
  - BCG240604 - Packaging Business Process Exit
  - BCG240607 - UnPackaging Business Process Entrance
  - BCG240608 - UnPackaging Business Process Exit
  - BCG240612 - Protocol Parse Business Process Entrance
  - BCG240613 - Protocol Parse Business Process Exit
Warning events
- BCG240605 - Packaging warning
- BCG240609 - UnPackaging warning
- BCG240614 - Protocol parse warning

Error events
- BCG240418 - Digest Generation Failure
- BCG240419 - Unsupported Signature format
- BCG240420 - Unsupported Signature algorithm
- BCG240606 - Packaging Error
- BCG240611 - Encryption failure
- BCG240610 - UnPackaging Error
- BCG210014 - Error Unpackaging Mime Message
- BCG240417 - Decryption failure
- BCG240424 - Insufficient message security error
- BCG240615 - Protocol parse error

The following events were created for use with custom senders:

Informational events
- BCG240616 - Sender Entrance
- BCG240617 - Sender Exit
- BCG250007 - Document Delivered

Warning events
- BCG240618 - Sender warning

Error events
- BCG250008 - Document Delivery Failed
- BCG250011 - First Delivery Attempt Failed
- BCG250012 - Delivery Retry Failed

You can also find this information in the product documentation, specifically in the WebSphere BI Connect Programmer Guide. Note that early versions of the Programmer Guide may include an underscore in the event code (for example BCG_250008), which is not correct.

Exceptions
The WebSphere BI Connect provides exception classes that extend the Java class Exception.

BCGException
BCGException is a base exception class, which is used by the user exit API. Some methods can throw exceptions. All the exceptions that can be thrown in the user exit API are derived from this class.
**BCGReceiverException**

BCGReceiverExceptions are raised when the receiver encounters errors and cannot generate an event. BCGReceiverExceptions can be thrown from several methods on the ReceiverInterface.

- `init()`
- `refreshConfig()`
- `startreceiving()`
- `stopreceiving()`

**BCGSenderException**

BCGSenderExceptions are raised when the sender encounters errors and cannot generate an event. BCGSenderExceptions can be thrown from the `init()` method on the SenderInterface.

**Tracing**

Event creation and application tracing are the main methods for problem determination in the WebSphere BI Connect system. As such, you must establish the tracing standards before beginning development. Distribute events through the code in their intended fashion. Make tracing calls to assist in troubleshooting and problem determination efforts. Some common rules of thumb for tracing:

- Create a trace message at the beginning and the end of each method.
- Create a trace messages at various special processing points.
- Perform tracing inside a catch block.

To create a trace statement, WebSphere BI Connect provides a `trace()` method on the BCGUtils class. This method has two signatures.

- Without exception object
  ```java
  public void trace(String severity, String category, String msg)
  ```

- With exception object
  ```java
  public void trace(String severity, String category, String msg, Throwable t)
  ```

To enable tracing in the WebSphere BI Connect system, you must modify the property files for the associated component. For receivers, you must modify the receiver-was.logging.properties file. You can find this file in the `<receiver_install_directory>\was\wbic\config\` directory.

For workflow components and senders, you must modify the router-was.logging.properties file. You can find this file in the `<router_install_directory>\was\wbic\config\` directory.
In either case, set the log4j.rootCategory property to debug and restart the associated server.

17.2.6 Business case test strategy

In Chapter 7, “Creating a basic B2B exchange” on page 121, we set up and deployed a solution, which described an interaction between Company A and Company E. This solution leveraged AS2 processing and closely mimics our document flow. The difference is that we will implement a new business package and custom extensions.

Our testing strategy takes an incremental approach and centers around this AS2 process flow. The goal is to slowly introduce custom components to a tested flow. By using the incremental approach, we isolate the impact of custom development as we proceed.

Our strategy is to:
1. Create a custom receiver and introduce this receiver to the existing AS2 process flow.
2. Create a custom sender and introduce this sender to the existing AS2 process.
3. Test the custom receiver and sender with AS2 documents.
4. Create the required workflow components (inbound, variable and outbound) and introduce these components to the existing process flow. This includes using the ITSO custom packaging and data formats, and therefore the flow no longer exchanges AS2 data.
5. Test the end-to-end solution including all aspects of the custom development effort.

Important: You must have a working, tested flow before you begin to introduce custom components. Otherwise it will be difficult to determine whether errors are due to a misconfiguration of WebSphere BI Connect or to the introduction of software bugs via the custom components.

17.2.7 Predeployment tests

We use JUnit to execute predeployment tests of the custom extensions. In some cases, predeployment tests are limited by the need to interact with the WebSphere BI Connect framework. In these cases, we use a post-deployment technique that leverages the strengths of WebSphere Studio Application Developer's debugging capabilities.
In the case of the sample solution, the business logic was built into a stand-alone framework that allowed us to test it separately using JUnit. This kept post-deployment testing to a minimum. We suggest this as a best practice since the develop, debug, and test cycle is significantly shortened when you do not have to constantly stop WebSphere BI Connect, redeploy your code, and restart WebSphere BI Connect.

17.2.8 Deployment

Before you can use the solution used, you must deploy the associated artifacts to the WebSphere BI Connect system. After the items are deployed, user and hub administrators can configure document flows using the newly developed components.

As part of the deployment process, the developer must create a few associated files to assist in the process. The files are the XML deployment descriptors that will be used by the console to provide the user with the correct and complete set of configuration properties. For example, when we create our custom receiver, we must also include an XML deployment descriptor. This deployment descriptor indicates to the WebSphere BI Connect Console’s user interface that, when a target is generated from the custom receiver, the user can set the necessary values for the target configuration such as the polling directory.

We cover deployment and provide practical examples of the steps required for deployment in the subsequent chapters.

17.2.9 Post-deployment tests

When debugging deployed code in the WebSphere BI Connect system, the developer has to configure the server and set up WebSphere Studio for Application Developers to listen for debug events.

Exporting custom components

Before you can begin testing, you must export the JAR files that represent the user exits and place the JAR in the following folders:

- `<install_wbic>\was\recevier\userexits`
- `<install_wbic>\was\router\userexits>`
Configuring the server
To configure the server, you must complete these actions:

1. Locate the server.xml file in the corresponding directories for the receiver, router and console:

   `<install>\*\was\config\cells\DefaultNode\nodes\DefaultNode\servers\server1

   Here, the asterisk (*) is either the receiver or the router.

2. Search for the debugMode parameter and change the value to true.

3. Set the address property to a unique value in each file. For example, if the address value is set to 7777 in the receiver server.xml file, set the address value in the router server.xml file to 7778. Using different debug ports is important if you want to connect simultaneously to both the router and the receiver for debugging.

   **Note:** As a reminder, Company A and E have installed all WebSphere BI Connect components on a single machine. Therefore, we need to make sure that router and receiver are configured for different debug ports.

4. Repeat this activity for both the Company A and Company E machines.

5. Restart the component instance.

Configuring WebSphere Studio Application Developer
After a WebSphere BI Connect Server is configured for debugging (see “Configuring the server” on page 520), it is possible to attach to it using WebSphere Studio Application Developer. In this way, you can debug code that is running on the server using source code stored elsewhere.
To begin, attach to the server.

1. Create a new Server project in WebSphere Studio Application Developer. Select File → New → Project. See Figure 17-7.

Figure 17-7 Starting a new server project
2. In the New Project window (Figure 17-8), select **Server** from the available project types in the left pane and select **Server Project** in the right pane. Click **Next**.
3. In the Create a New Server Project window (Figure 17-9), give the project an appropriate name and click **Finish**.

![Create a New Server Project window](image1.png)

*Figure 17-9  Configuring the Server project name and location*

4. If you see the Confirm Perspective Switch (Figure 17-10), and are prompted as to whether you want to switch to the Server perspective, choose **Yes**. If not asked, switch to the Server perspective manually.

![Confirm Perspective Switch window](image2.png)

*Figure 17-10  Prompt to switch to the Server perspective*
You should now see the server project that you created in the Server perspective as shown in Figure 17-11.

Figure 17-11   Server perspective with a newly created Server project
Next, you create the individual server configurations.

1. In the Server Configuration view, right-click **Servers**. Choose **New → Server and Server Configuration**.

2. In the Create a New Server and Server Configuration window (Figure 17-2), type a descriptive name for the server. For Server type, under the heading WebSphere version 5.1, choose **Express Server Attach**. Click **Next**.

![Create a New Server and Server Configuration](image)

**Figure 17-12 Configuring the new Server configuration**
3. In the Server Attach Instance panel (Figure 17-13), type the host name for the WebSphere BI Connect server to which you are connecting. Click **Next**.
4. In the WebSphere v5 Server Attach Configuration panel (Figure 17-14), change the debug port to match the port you set in “Configuring the server” on page 520. Click **Finish**.

*Figure 17-14  Configuring the debug port for the server*
5. You now see your server configuration in the Server Configuration view (Figure 17-15). Repeat this process for the server you want to debug in your custom development. For our project, we create profiles for the Receiver and Router on both Company A and E (total of four profiles).

Figure 17-15  New server configuration in the Server perspective
Debugging with WebSphere Studio Application Developer

To debug deployed code, after configuring WebSphere Studio Application Developer, ensure that appropriate breakpoints are set.

1. Set the breakpoints. Either double-click to the left of the line of code where you want to break. Alternatively, right-click to the left of the line of code where you want to break and choose Add/Remove Breakpoint.
2. After you configure the breakpoints, begin debugging. Switch to the Debug perspective, and choose the Server view tab. Right-click the server you want to debug and select Debug.

Figure 17-17  Attaching to a server
3. When the server reaches your code, you are asked to supply the folder that contains the source in the Debugger Source Lookup window (Figure 17-18).

![Figure 17-18  Source lookup dialog](image-url)
a. Deselect the option **Use default source lookup path** to enable the choices on the right as shown in Figure 17-19.

b. Click **Add Projects**.
4. In the Project Selection window (Figure 17-20), select your project from the list. Click OK.

*Figure 17-20  Adding the relevant project for source code lookup*
The code now breaks at your first breakpoint and highlights that line as shown in Figure 17-21.

![Debugging code running remotely on the attached server](image)

**Figure 17-21** Debugging code running remotely on the attached server

Use the WebSphere Studio Application Developer debug commands such as Step Into, Step Over and Step Return to debug the code normally.

You can find more general information about working with WebSphere Studio Application Developer or with WebSphere Application Server - Express in the following Redbooks:

- **WebSphere Studio Application Developer Version 5 Programming Guide**, SG24-6957
- **WebSphere Application Server - Express V5.0.2 Developer Handbook**, SG24-6555
Implementing receiver and sender user exits

This chapter discusses the behavior of senders and receivers in general and helps you apply that knowledge by building a custom receiver and sender. The custom sender and receiver are a variation of the standard file system receiver and sender. This discussion enables you to focus on the mechanics of writing a sender and receiver, and not on specific transport details.
18.1 Updating WebSphere BI Connect

The receiver and sender user exit discussed in this chapter were developed and
tested in an upgraded WebSphere BI Connect environment. The installation of
Fix Pack 3 is required for these exits to work correctly. You can find details about
Fix Pack 3 and other support information on the Web at:

18.2 The frameworks and building blocks

Receivers and senders are used to interact with transports such as HTTP, Java
Message Service (JMS), and SMTP. To interact with the rules of document flow
within the existing WebSphere BI Connect system, WebSphere BI Connect
provides a framework for developing custom transports. This framework for
receivers and senders abstracts the details of the interaction with WebSphere BI
Connect and allows developers to create a number of extensions by
implementing Java classes. The framework provides a tremendous benefit.
Through the implementation of the existing framework interface, WebSphere BI
Connect manages many of the difficult aspects of B2B integration.

The framework provides:

- A single interface for transport development
- Persistence of a document
- Non-repudiation
- The ability for the developer to focus on the development of the transport
  interactions

18.2.1 Transport development

We discuss transport development in two sections:

- Receiver development
- Sender development

Our business case requires business document exchange between Company A
and Company E. This document flow includes integration to applications that use
a custom file system sender and a custom file system receiver. To meet this
requirement and demonstrate user exit functionality, we design a custom receiver
and sender to interact with our file system. We choose a file system as the
transport because file system transports to and from the applications do not
require you to install and deploy and extraneous prerequisites such as JMS or
FTP. Our custom receiver and sender are developed using their respective
frameworks and APIs.
18.3 The receiver

Before we explain the design and implementation of the custom receiver, let us review our receiver requirements and re-introduce the receiver concept.

Receivers are entry points to the WebSphere BI Connect system. They are responsible for interacting with a transport to accept new inbound business documents.

In our design requirements for the receiver, the custom file system receiver must:

- Poll a configured directory structure looking for files that match a configured extension.
- Retrieve files that match the extension and directory and submit those documents for inbound WebSphere BI Connect processing.
- Provide debugging capability and raise any event and errors that are encountered in the retrieval of documents as appropriate. Archive failed documents and create internal WebSphere BI Connect events that represent those activities.
- The receiver is not responsible for unpackaging or decrypting the document. Unpackaging and decryption are covered in Chapter 19, “Implementing workflow user exits” on page 587.

To relate this activity to the complete solution flow, refer to Figure 18-1, which illustrates the area of focus for this discussion.

![Figure 18-1 Custom receiver monitoring a specified directory on the file system](image-url)
18.3.1 Receiver processing flow

Regardless of the receiver requirements, there are common tasks that receivers must perform.

1. Detect message arrival on the transport.
   Receivers use one of two methods to detect request message arrival:
   - Polling the targets defined for this transport, as the provided JMS receiver does
   - Receiving callbacks from the transport, as the provided HTTP/S receiver does

2. Retrieve the message from the transport.
   The receiver retrieves the request message and any transport attributes, such as headers, from the transport. This may require the creation of temporary files on the file system.

3. Generate WebSphere BI Connect required headers.

   **Note:** You may use the steps 4 and 5 in either order.

4. Special WebSphere BI Connect defined metadata is used for further processing of the document. Receivers may create one or more of these metadata that are appropriate, from transport message attributes or in other ways. The list of headers consists of:
   
   - BCGDocumentConstants.BCG_RCVR_DESTINATION: This attribute is set by Receivers on ReceiverDocumentInterface object, upon receiving the document from a target. This target is associated with destination type as “production”, “test”, and so on. The destination type configured for the target can be read from the receiver configuration using the BCGDocumentCosntants.BCG_TARGET_DESTINATION attribute.
   
   - BCGDocumentConstants.BCG_RCVD_IPADDRESS: This is the host IP address where the document is received.
   
   - BCGDocumentConstants.BCG_INBOUND_TRANSPORT_CHARSET: The character set is obtained from the transport headers.
   
   - BCGDocumentConstants.BCG_INBOUND_CHARSET: The character set is used for the inbound document.
   
   - BCGDocumentConstants.BCG_REQUEST_URI: This is the URI of the target on which the request is received.
   
   - BCGDocumentConstants.BCG_RCVD_DOC_TIMESTAMP: This indicates the received document time.
- **BCGDocumentConstants.BCG_RCVD_CONTENT_LENGTH**: This indicates the size of the received content.

- **BCGDocumentConstants.BCG_RCVD_MSG_LENGTH_INC_HDRS**: This header indicates the size of the received content including the headers.

- **BCGDocumentConstants.BCG_RCVD_CONTENT_TYPE**: This indicates the content type of the request.

The receiver request document, which is forwarded to Document Manager for further processing, consists of the transport message, transport headers, and these WebSphere BI Connect headers.

5. **Perform preprocessing.**

The receiver calls a WebSphere BI Connect component, the Receiver Framework, to execute the preprocessing. The Framework moves through its configured list of handlers, either product supplied or user defined, until one that can handle the document is located. The document is processed. This step may return one or more documents, and all receivers must be designed to handle multiple returns.

6. **Check the sync status.**

The receiver calls the Receiver Framework to execute sync check. The Framework moves through its configured list of handlers, either product supplied or user defined, until an appropriate one is located. The outcome of this step determines the receiver's next step. If the request is asynchronous, path A is followed. If the request is synchronous, path B is followed.

7. **Process an asynchronous request.**

If the request is asynchronous (does not require a response document to be returned to the originating trading partner), the receiver calls the Framework to process the request document. The Framework handles storing the information in a place from which Document Manager will retrieve it.

8. **Process the synchronous request.**

If the request is synchronous (requires a response document to be returned to the originating trading partner), the receiver calls the Framework to process the request document. There are two possible types of synchronous requests: blocking and non-blocking. In **blocking mode**, the receiver's calling thread is blocked until the Framework returns the response document to it from Document Manager. In **non-blocking mode**, the receiver's calling thread returns immediately. When the Framework receives the response document at a later time, it calls the `processResponse` method on the receiver to pass the response document back. A correlation object is used to synchronize the originating request with this response.

In the case of a synchronous request, the receiver calls the Framework to execute postprocessing on the response document before it is returned to the originating partner. The Framework moves through its configured list of handlers, either product supplied or user defined, until an appropriate one is located. The document is processed.

10. Complete the processing.

In case of a synchronous request, the response document is returned to the originating trading partner over the transport. The receiver calls the `setResponseStatus` method on the Framework to report on the success or failure of the response delivery. The receiver then removes the request message from the transport.

For the purpose of our business case, we do not support synchronous requests. Therefore the steps involving synchronous request and response processing are out of the scope of this discussion.

18.3.2 The receiver framework

WebSphere BI Connect provides the APIs and the class definitions for the developer to implement a custom receiver. Using the Receiver Framework, a custom receiver can be developed for unsupported customer requirements. We discuss the class and interface definitions that WebSphere BI Connect provides and relate the business case to our implementation of a custom file system receiver. As shown in Figure 18-2, the custom receiver interacts with the transport. The Receiver Framework acts as an intermediary between custom receivers and the WebSphere BI Connect system as a whole.

![Figure 18-2 Custom receiver interacting with the Receiver Framework](image-url)
**ReceiverConfig**
The ReceiverConfig class is provided to allow configuration parameters to be passed to new receivers on initialization.

**Receiver requirements and configuration properties**
Our receiver requires configured values to function properly. To meet our business requirements, our team has designed a custom receiver that functions as we explain in the following paragraphs.

The receiver polls a configured directory structure looking for new files. These files contain a configured extension. This functionality requires two attributes:

- **File extension**: What is the desired file extension? We develop this receiver under the assumption that customer implementation teams may want to scale the solution by using file extensions to load balance the receivers. A receiver can be configured to look for a particular file extension.

- **Directory root path**: In which file system directory should the receiver look for inbound files?

To manage the polling functionality of the file system receiver, we define two additional configuration receiver properties.

- **Poll interval**: This property helps to determine how often the receiver should poll the configured directory looking for new files.

- **Thread number**: How many threads should be created to process the incoming files? This property is currently not implemented in the sample receiver exit. However, it should be considered for a production implementation of a receiver, so that you can control the level of parallelism in the receiver.

In addition to increasing parallelism by creating additional threads, you can consider a parameter to bloc the amount of work that is passed to WebSphere BI Connect. For example, consider a maximum number of files that can be read by the exit and passed to WebSphere BI Connect per polling interval.

Now that you understand which parameters are required, the question remains as to how a user configures the custom receiver and passes those configured values to the receiver on initialization. As you will see, in the `init()` method, the framework provides those values as configuration objects, but this explanation requires a more clarity.

For each WebSphere BI Connect extension, a developer must create a deployment descriptor. Figure 18-3 shows the descriptor used for the receiver exit.
The deployment descriptor is used by the WebSphere BI Connect console to display the configuration options for the extension. In this case, our receiver has four values in the deployment descriptor.

- Poll Interval
- Directory to Poll
- File Extension
- Thread Nbr

The console reads this deployment descriptor and provides configuration text boxes that allow the user to create a target. Each target configuration is stored in a Config object. When the `init()` method on the ReceiverInterface is called by the framework, these configured values (one or more Config objects) are passed to the exit.
We call the `getTargetConfigs()` method to return the configuration values for all targets for that receiver. Each `Config` object contains the values provided for the Poll Interval, Directory to Poll, and so on. We then use the `getAttribute` method, passing in the String value of the attribute name, to obtain the configured value. Figure 18-4 shows how the `Config` object is interrogated for the attribute Directory to Poll.

```java
inDirPath = (String)targetConfig.getAttribute(ExitConstants.DIR_TO_POLL);
```

Figure 18-4  Getting the value of an attribute of a configured target
**ReceiverInterface**

The development of the ReceiverInterface class represents 90% of the new receiver effort. The receiver interface class implements the receiver's interaction with the custom transport. This is where the event notification mechanism runs and the polling for new documents to process is implemented. This is a new file system transport that uses rules for exchanging documents through a configured directory structure and extension.

**ReceiverInterface methods to implement**

When developing a custom receiver, the developer must create a class that implements ReceiverInterface. In our case, we created the class in Figure 18-5.

```java
public class FileMatchingReceiver implements ReceiverInterface {
}
```

*Figure 18-5 Custom receiver class*

The developer must create the code that interacts with the transport and submit the inbound documents for processing. We show you how we implemented these methods for our business case. The developer must use the following methods:

- **init()**: This method (Figure 18-6) is called by the framework when the receiver is started. We initialize our variables and consume the target configuration provided by the framework.

```java
public void init(Context context, ReceiverConfig receiverConfig)
    throws BCGReceiverException {
    bcgUtility = new BCGUtil();
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        "FileMatchingReceiver",
        "Entering init()");

    this.context = context;
    this.receiverConfig = receiverConfig;
    fmrThreads = new ThreadGroup("File Matching Receiver Threads");
    currConfigs = new LinkedList();
    currConfigs.addAll(receiverConfig.getTargetConfigs());
    transportType = receiverConfig.getTransportType();

    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        "FileMatchingReceiver",
        "Leaving init()");
}
```

*Figure 18-6 The init() method*
- **refreshConfig()**: This method is used to update the configuration values of the targets. The information is passed to the receiver by the framework when the framework detects changes to the configuration. In our implementation of this method, we check the values of the new configuration object and compare them to the existing values. If a difference is detected during that compare process, the receiver replaces the configuration.

- **startReceiving()**: We first ask our main class (FileMatchingReceiver) to receive the target configuration in the `init` method. With the target configuration, we can start an instance of the receiver for each target configuration provided. This target configuration is used as an input parameter. In this method, we initialize the new thread and start the thread processing for that the target configuration. The thread continues to process until told to do otherwise.

A target can be enabled or disabled from the Console. Thus, strictly speaking, the target should first read its status before proceeding, which is not implemented in the code snippet in Figure 18-7. The configuration attribute that holds the status is BCGDocumentConstants.BCG_TARGET_STATUS.

```java
public void startReceiving() throws BCGReceiverException {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                    "FileMatchingReceiver",
                    "Entering startReceiving()");
    List currConfigs = this.receiverConfig.getTargetConfigs();
    Iterator currConfigsIter = currConfigs.iterator();

    while(currConfigsIter.hasNext()) {
        Config currConfig = (Config)currConfigsIter.next();
        startNewFMRThread(currConfig);
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                    "FileMatchingReceiver",
                    "Leaving startReceiving()");
}
```

*Figure 18-7  startReceiving method*

- **stopReceiving()**: This method (Figure 18-8) is called by the framework when the receiver should stop receiving documents. This method can be called for several reasons including:
  - An internal component requests the receiver to terminate.
  - The receiver is disabled or removed from the console.
  - The receiver has requested itself to be removed by calling the `remove()` method provided by the framework. The framework then calls the `stopReceiving()` method.
When this method is called, our receiver enumerates through the thread group that was created during initialization and attempts to terminate each thread.

```java
public void stopReceiving() throws BCGReceiverException {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
    "FileMatchingReceiver",
    "Entering stopReceiving()");
    int numThreads = fmrThreads.activeCount();
    Thread [] threads = new Thread[numThreads];
    fmrThreads.enumerate(threads);
    for(int i = 0; i < threads.length; i++) {
        FileMatchingReceiver.FileMatchingReceiverThread fmrt;
        fmrt = (FileMatchingReceiver.FileMatchingReceiverThread) threads[i];
        fmrt.requestStop();
        int count = 0;
        while(fmrt.isAlive() && count++ < 10){
            try {
                fmrt.join();
            } catch (InterruptedException ie) {
                // intentionally left blank
            }
        }
        if (fmrt.isAlive()) {
            String currTarget = fmrt.getTargetConfig().getName();
            throw new BCGReceiverException(
                "Target: " + currTarget + ", ThreadID: " +
                fmrt.getName() + ": " + "Unable to stop thread";
            )
        }
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
    "FileMatchingReceiver",
    "Leaving stopReceiving()");
}
```

*Figure 18-8  stopReceiving method*

- **processResponse()**: This method is used to process the response of documents that were submitted synchronously. We do not support synchronous processing in this version of the sample. This method is not used.

**Receiver threading**

We use a threading pattern to create each instance of the receiver. We implement the `run()` method for our thread and start the polling process. We use a while loop and management methods to control the polling behavior. On each
loop pass, the receiver examines the file system, looking for the configured documents.

When a document is found, the receiver must create a instance of the ReceiverDocumentInterface class. This instance is eventually passed to the WebSphere BI Connect system for document processing. It contains all the information such as transport metadata, WebSphere BI Connect metadata, and a reference to the actual business document.

The poll() method uses the parameters that were provided by the target configuration and searches the file system for files matching our configuration. If a match is detected, the method returns the matching files.

The processDocument() method is where our receiver interacts with the receiver framework. In this method, we perform the following actions:
1. Instantiate an instance of the framework (see Figure 18-9).

```java
ReceiverFrameworkInterface rcvFramework
    = BCGReceiverUtil.getReceiverFramework();
```

*Figure 18-9  Getting the ReceiverFrameworkInterface instance*

Call the receiver framework to execute any configured handlers. The code snippet in Figure 18-10 represents a call to the preProcess handlers.

```java
ReceiverDocumentInterface requestDocs[]
    = rcvFramework.preProcess(transportType,target,request);
```

*Figure 18-10  Calling the preProcess handlers*
2. Submit the document or documents to the framework for processing. There are multiple signatures for the process() method. Depending on the type (asynchronous, synchronous blocking, or synchronous non-blocking), call the correct process() method (see Figure 18-11) to submit the inbound document for document processing.

```java
boolean sync = rcvFramework.syncCheck(transportType,target,request);
if(!sync){
    rcvFramework.process(transportType,request);
} else {
    throw new BCGReceiverException(target + "+ getName() + ": Sync check return true for non-synchronous target");
}
```

Figure 18-11  Call to the process() method

In our case, we call the syncCheck method. Based on the response, a call is made to the process() method for asynchronous processing.

As previously mentioned, our polling behavior is defined by loops in the run() method and a few methods to assist in the receiving process. We created the following methods for our thread that represent the management of the thread:

- requestStop()
- requestPause()
- endPause()

Notice that the values that the thread uses for management are checked at various points of the run() method. These methods work in conjunction with the stopReceiving() method from the ReceiverInterface and are implemented by our FileMatchingReceiver class.

Finally, the configOK() method is used to validate the target configurations provided. This method ensures that the parameters being passed in are legitimate for our purpose. For example, if the directory provided was not a valid directory path, the configOK routine would return the error.

**ReceiverDocumentInterface**

When the receiver receives a document, the receiver creates an instance of the ReceiverDocumentInterface class. This class is vitally important because it represents all of the information known about the inbound document. This class contains methods by which the receiver developer may set a variety of metadata about the inbound request.
There are two types of metadata.

▶ Transport headers

Transport headers can be set by the developer depending on the information that is required further downstream in the document processing. Some examples of transport headers are:

- `BCGDocumentConstants.BCG_RCVD_IPADDRESS`
- `BCGDocumentConstants.BCG_REQUEST_URI`

Transport headers are user defined. The receiver developer creates the transport headers and stores them as key/value pairs in a HashMap object.

We use `setTransportHeader()` and `getTransportHeaders()` to manipulate or determine the values of the transport headers.

▶ WebSphere BI Connect headers

WebSphere BI Connect headers (also known as BCG headers) are used for document processing. These headers are read by the document processing engine at run time and the appropriate workflows are called. Some examples of WebSphere BI Connect headers are:

- `BCG_FRBUSINESSID`: The identifier for the sending participant
- `BCG_TOBUSINESSID`: The identifier for the receiving participant
- `BCG_FRPROTOCOLCD`: Indicates the business packaging protocol being used

WebSphere BI Connect headers are manipulated using the `getAttribute` and `setAttribute` methods on the ReceiverDocumentInterface.

For our example, we set most of the WebSphere BI Connect required document headers in the Fixed Inbound Workflow phase of WebSphere BI Connect document management. This is addressed in Chapter 19, “Implementing workflow user exits” on page 587.

Our custom receiver populates the metadata for the transport and the BCG-specific attributes as available.

The receiver developer also sets a reference to the inbound business document. Only a reference is used at this point. The actual file retrieved from the file system resides in a temporary location after detection.

After the metadata and the inbound file reference are set, the instance of the ReceiverDocumentInterface class contains all of the necessary information to begin document processing. The instance of the ReceiverDocumentInterface class is then ready to be submitted to the framework for processing.
**ReceiverFrameworkInterface**

This interface represents the main APIs for the development kit. This interface provides the methods for the receiver developer to interact with the WebSphere BI Connect document management system.

The main method for the receiver framework is the `process()` method. As noted, this method has three signatures for different processing types:

- Asynchronous
- Synchronous blocking
- Synchronous non-blocking

When the `process()` method is called, the framework writes the request document into the WebSphere BI Connect internal file system.

Beyond the `process()` method, several other APIs are provided. The `remove()` method is provided for receiver management. The receiver should invoke the `remove()` method to perform a managed shutdown. This prompts the framework to call the `stopReceiving()` method on the receiver class and stop inbound document processing.

Developers can create custom handlers to modify the behavior of an existing receiver. User-defined receivers also leverage this flexibility. The custom receiver developer enables this functionality by using the following APIs:

- PreProcess
- SynchCheck
- PostProcess

As we saw, these methods are called by the custom receiver. When called, the receiver framework iterates down the configured handler chain and executes the appropriate handler.

The final API for the framework is the `SetResponseStatus()` call. This method allows the developer to tell the framework the outcome of a synchronous request after the originator has received the response. For example, assume an HTTP response is provided during the document flow. After the HTTP response is provided to the original requestor, the receiver developer calls this method to indicate to the framework whether the response was returned successfully.
**BCGReceiverUtil**
Throughout our explanation of our custom receiver, you may have noticed the various calls to the BCGReceiverUtil class. BCGReceiverUtil is a static class that provides various essential functions to the receiver developer.

The `createReceiverDocument()` method creates the instance of the `ReceiverDocumentInterface` class. The `getReceiverFramework()` method returns the receiver framework so that document processing may begin.

The BCGReceiverUtil also provides access to file processing through the use of:
- `getTempDir()`
- `getRejectDir()`
- `getArchiveDir(String receiverType, String targetName)`

These methods are important for business document management and storage.

**18.3.3 WebSphere BI Connect provided receiver and custom handlers**

Although not required by our use case, you can create new handlers to be used with a WebSphere BI Connect provided receiver. You can add custom handlers to the system by using the provided configuration points. The WebSphere BI Connect console is used to configure these custom handlers. The handler must be added to the system by creating a deployment descriptor based on the handler type. Valid handler types for receivers are:

- RECEIVER.PREPROCESS.FtpDirectory
- RECEIVER.SYNCCHECK.FtpDirectory
- RECEIVER.SYNCRESPONSEPROCESS.FtpDirectory
- RECEIVER.PREPROCESS.JMS
- RECEIVER.SYNCCHECK.JMS
- RECEIVER.SYNCRESPONSEPROCESS.JMS
- RECEIVER.PREPROCESS.Smtp
- RECEIVER.SYNCCHECK.Smtp
- RECEIVER.SYNCRESPONSEPROCESS.Smtp
- RECEIVER.PREPROCESS.HttpS
- RECEIVER.SYNCCHECK.HttpS
- RECEIVER.SYNCRESPONSEPROCESS.HttpS
- RECEIVER.PREPROCESS.FileDirectory
- RECEIVER.SYNCCHECK.FileDirectory
- RECEIVER.SYNCRESPONSEPROCESS.FileDirectory

The schema for the deployment descriptors can be found on the installation CD of the product.
When deploying a custom Receiver Handler, use these steps:

1. Navigate to the target that you want to configure. Select **HubAdmin → Hub Configuration → Targets**.

2. Click the target from the list provided. This displays the current target attributes.

3. Edit the target.
   a. Use the drop-down list to select the configuration point that you want to use.
   b. Select the imported user-defined receiver handler from the available list.
   c. Click **Move** to add the handler to the configured handler chain.
   d. Use the buttons provided in the Console to move the handler up or down in the handler chain as desired.

### 18.4 The sender

Before we explain the design and implementation of the custom sender, let us review our sender requirements and re-introduce the sender concept. Senders are exit points for the WebSphere BI Connect system. They are responsible for interacting with a transport to send processed outbound business documents.

Let us clarify our design requirements for the sender. The custom file system sender must:

- Create files on a file system within a configured directory and with a configured extension.
- Provide debugging capability and raise any errors that are encountered during the creation of outbound documents and create internal WebSphere BI Connect events that represent those activities.
- The sender is not responsible for packaging or encrypting the document. Packaging and encryption are covered in Chapter 19, “Implementing workflow user exits” on page 587.

To relate this activity to the complete solution flow, refer to Figure 18-12, which illustrates the area of focus for this discussion.
18.4.1 Sender processing flow

Regardless of the sender requirements, senders must perform the following common tasks:

1. Receive the document.
   
   The business document to be processed is placed in the gateway input directory.

2. Perform any preprocessing.
   
   The Sender Framework, a WebSphere BI Connect internal component, is invoked. The Framework moves in order through its Console configured list of handlers until one, either WebSphere BI Connect delivered or user-defined, that can process the document is located. The document is processed.

3. Initialize the sender.
   
   The Framework calls the sender's `init()` method.

4. Send the document.
   
   The Framework calls the sender's `send()` method. The sender creates a `SenderResult` object to store transmission and status information. Then it sends the message, using the destination specified in the gateway, or, by preference, in the message itself, should it contain that information.

5. Store the response document.
   
   In the case of a synchronous request, the sender writes the response document to a file and sets the File object in the `SenderResult` object.
6. Perform any postprocessing.

The Framework moves through its Console configured list of handlers until an appropriate one, either WebSphere BI Connect supplied or user-defined, is located. The SenderResult object is processed.

7. Complete the processing.

The request document is removed from the gateway queue. In the case of a synchronous request, the response document is introduced into the system and flows through the system like any other business document.

For the purpose of our business case, we do not support synchronous requests. Therefore, the steps involving synchronous response processing are outside the scope of this discussion. Additionally, we do not use custom preprocessing and postprocessing handlers in our design.

18.4.2 The sender framework

WebSphere BI Connect provides the APIs and class definitions for the developer to implement a custom sender. The framework for sender development provides a consistent means for interacting with WebSphere BI Connect document flows. Sender development is different from Receiver development. Recall that in our custom Receiver code, we called the framework to iterate through the configured handlers. See Figure 18-13.

```java
ReceiverDocumentInterface requestDocs[]
    = rcvFramework.preProcess(transportType,target,request);
/* sync check is not required in our case, but is included here
 * as best practice so that other handlers can be configured.
 */
boolean sync = rcvFramework.syncCheck(transportType,target,request);
if(sync){
    rcvFramework.process(transportType,request);
}
```

*Figure 18-13 Custom Receiver calling Receiver Framework*

Senders are different in that the framework calls the preprocess and postprocess handlers, *not* the custom sender. The preprocess handler and postprocess handlers act as “book ends” to the main custom sender class that implements SenderInterface. See Figure 18-14.
We are not required to create preprocess and postprocess handlers for our business case.

**Config**

The generic Config class is provided to allow configuration parameters to be passed to new senders on initialization.

**Sender requirements and configuration properties**

Our sender requires configuration values to function properly. To meet our business requirements, our team designed a custom sender that functions as follows. The sender creates files in a configured directory structure. These files are created with a configured extension. This functionality requires two attributes:

- **File extension**: What file extension is desired on the output file?
- **Directory root path**: In which file system directory should the sender create the outbound files?
For each WebSphere BI Connect extension, a developer must create a deployment descriptor (Figure 18-15). The WebSphere BI Connect console uses the deployment descriptor to display the configuration options for the extension. In this case, our sender has two values in the deployment descriptor.

- File Extension
- Document Root Path

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:GatewayDefinition
xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance"
  <tns:GatewayClassName>com.ibm.itso.wbic.exits.FileNamingSender</tns:GatewayClassName>
  <tns:Description>Writes final file name based on document details</tns:Description>
  <tns:TransportTypeName>File Naming Sender</tns:TransportTypeName>
  <tns:TransportAttributes>
    <tns2:ComponentAttribute>
      <tns2:AttributeName>File Extension</tns2:AttributeName>
    </tns2:ComponentAttribute>
    <tns2:ComponentAttribute>
      <tns2:AttributeName>Directory Root Path</tns2:AttributeName>
    </tns2:ComponentAttribute>
  </tns:TransportAttributes>
</tns:GatewayDefinition>
```

*Figure 18-15  Deployment descriptor for the custom sender*

The console reads this deployment descriptor and provides configuration text boxes that allow the user to create a gateway. Each gateway configuration is stored in the Config object. When the `init()` method on the SenderInterface is called by the framework, these configured values (the Config object) are passed to the exit using the Config class.

**SenderInterface**

The SenderInterface describes the interaction with the transport. This is where the sender interacts with the transport when sending business documents. This is a new file system transport that uses rules for exchanging documents through a defined directory structure. The function of the sender is the exact opposite of the receiver.
SenderInterface methods to implement

When developing a custom sender, the developer must create a class that implements SenderInterface. In our case, we created the class shown in Figure 18-16.

```java
public class FileMatchingSender implements SenderInterface {
}
```

Figure 18-16  Custom sender class

The developer must create the code that interacts with the transport and create new files on the file system. We show you how we implemented these methods for our business case. The developer must use the following methods to do so.

- **init()**: The `init()` method (Figure 18-17) is called by the framework when the sender is started. We initialize our variables and consume the gateway configuration provided by the framework. Within the `init()` method, we preserve the inbound parameters and validate the configuration parameters.

  The `configOK` method is the private method that validates the configuration parameters. If the validation fails, the `init()` method throws a `BCGSenderException`. 

public void init(Context context, Config deliveryConfig) throws BCGSenderException {

    try {
        bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
            this.getClass().getName(),
            "Entering initialization.");

        // Preserve original configuration
        this.context = context;
        this.deliveryConfig = deliveryConfig;

        // check the configuration parameters
        boolean success = configOK();

        if (!success){
            bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_ERROR,
                this.getClass().getName(), "Config parameters failed: " +
                failedReason);

            throw new BCGSenderException();
        }

        bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
            this.getClass().getName(),
            "Exiting initialization.");
    }

    catch (Exception e) {
        throw new BCGSenderException();
    }
}

Figure 18-17  The init() method

- send(): The send() method (Figure 18-18) is the main method for the SenderInterface. The Sender Framework invokes the send() method after initialization. Within the send() method, the developer must interact with the transport. In our business case, the sender must create a new file in the configured directory with the configured extension. The file name is derived from a timestamp and the attribute BCG_FRBUSINESSID that is found in the actual document. Company E’s back-end application picks up the new files for processing.
public SenderResult send(BusinessDocumentInterface document) {
    try {
        bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                      "FileNamingSender", "Entering send() method.");

        // Retrieve document information.
        this.document = document;
        currentDocUUID = document.getDocumentUUID();

        // create the event object and log the event
        String[] params = {"com.ibm.itso.wbic.exits.FileNamingSender"};
        EventInfo ei = new EventInfo("BCG240616", currentDocUUID, params);
        result.addEvent(ei);

        // create new destination file name using time stamp,
        // document path and file ext from config
        String destName = new String(docPath);
        if (!destName.endsWith(System.getProperty("file.separator"))){
            destName = destName + System.getProperty("file.separator") +
            document.getAttribute(BCGDocumentConstants.BCG_FRBUSINESSID)+ fileExtension;
        }
        long currTime = System.currentTimeMillis();
        Long tempTimeStamp = new Long(currTime);
        destName = destName + tempTimeStamp.toString() +
                  document.getAttribute(BCGDocumentConstants.BCG_FRBUSINESSID)+ fileExtension;
        bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                      "FileNamingSender", "File name created: " + destName);

        // create destination file
        File destFile = new File(destName);

        // get a reference to the document
        File currFile = document.getDocument();
    }
}
// try writing the document file to the destination directory
if (currFile != null) {
    success = currFile.renameTo(destFile);
}

if (!success) {
    // create an error event
    String message = "Sender failed to send: Creation of file or rename failed";
    params[0] = message;
    // create the event object and log the error
    EventInfo e1 = new EventInfo("BCG250008", currentDocUUID, params);
    result.addEvent(e1);

    // set sender result state failed!
    result.setSendStatus(BCGDocumentConstants.BCG_SENT_STATUS_FAILED);
} else {
    // create an event that indicates success
    String message = "Sender was sent successfully! New file name is " + destName;
    params[0] = message;

    // create the event object and log the event
    EventInfo e1 = new EventInfo("BCG250007", currentDocUUID, params);
    bcgUtil.logEvent(e1);

    // set SenderResult state success!
    result.setSendStatus(BCGDocumentConstants.BCG_SENT_STATUS_SUCCESS);
} else {
    bcgUtil.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
       this.getClass().getName(),
       "Current business document is null!");
}
Figure 18-20  The send() method (Part 3 of 3)

- **cancel():** The sender framework provides the `cancel()` method for the sender developer to implement. The sender framework calls this method to indicate that the sender should discontinue processing of the current document. This method is especially useful when the sender was developed with a “retry” functionality. This allows the framework to cancel the retry. In our business case, we do not use this method because there is no requirement for a retry.
SenderResult
The SenderResult object is used by the sender to communicate send status and coordinate response processing in the case of a synchronous send. Response information and status are set during the course of the send() method. That is, the send() method returns an object of type SenderResult. The SenderResult object is used to pass this information to and from the custom sender and the WebSphere BI Connect system. See Figure 18-21.

The following statements provide the possible values for the send status:

- BCG_SENT_STATUS_SUCCESS: Indicates a successful send
- BCG_SENT_STATUS_FAILED: Indicates the sending of the business document failed

```java
if(!success) {
    ...
    result.setSendStatus(BCGDocumentConstants.BCG_SENT_STATUS_FAILED);
} else {
    ...
    result.setSendStatus(BCGDocumentConstants.BCG_SENT_STATUS_SUCCESS);
}
...
return result;
```

*Figure 18-21  Setting the SenderResult status and returning the object*

18.4.3 WebSphere BI Connect provided sender and custom handlers

Although not required by our use case, you can create new handlers to be used with a WebSphere BI Connect provided senders. You can add custom handlers to the system by using the provided configuration points. The WebSphere BI Connect Console is used to configure these custom handlers. You must add the handler to the system by creating a deployment descriptor based on the handler type.

The valid handler types for senders are:

- GATEWAY.PREPROCESS.Http1.1
- GATEWAY.POSTPROCESS.Http1.1
- GATEWAY.PREPROCESS.Https1.0
- GATEWAY.POSTPROCESS.Https1.0
- GATEWAY.PREPROCESS.Https1.1
- GATEWAY.POSTPROCESS.Https1.1
- GATEWAY.PREPROCESS.Ftp
- GATEWAY.POSTPROCESS.Ftp
You can find the schema for the deployment descriptors on the installation CD of the product. When deploying a custom sender handler, use these steps:

1. Navigate to the participant that has the configured gateway that you want to modify. Click the gateway from the list provided.
2. You see the current gateway attributes. Edit the gateway attributes.
   a. Use the drop-down list to select the configuration point that you want to use.
   b. Select the imported user-defined sender handler from the available list.
   c. Click Add to add the handler to the configuration chain.
   d. Click Move Up and Move Down to configure the handler in the correct position.

18.5 Deploying and testing the receiver and sender

You need to deploy the custom receiver and sender for the ITSO project to the existing document flow between Company A and Company E. The following section explains how these components are introduced into the WebSphere BI Connect system. The instructions walk you through the necessary steps.

To continue our description of the Company A to Company E flow, we start our configuration of the Company A WebSphere BI Connect instance (custom receiver) and move on to the required configuration for Company E (custom sender). After we complete the required steps, we run an end-to-end test.
18.5.1 Creating the user exits JAR file

The ITSO user exit code must be distributed to both Company A and Company E. Follow these steps:

1. Distribute the JAR file. Export the project from WebSphere Studio Application Developer. Right-click the project in WebSphere Studio Application Developer and select **Export**. See Figure 18-22.
2. In the Export window (Figure 18-23), select **JAR file** and click **Next**.

![Figure 18-23   Selecting the JAR file](image)
3. In the JAR Export window (Figure 18-24), select the packages and source files that implement the exit. Provide a name and directory for the new JAR file. Click **Finish**.

**Note:** Figure 18-24 lists packages and source files for more than the sender and receiver exits. It also includes classes and packages that are discussed in Chapter 19, “Implementing workflow user exits” on page 587.
18.5.2 Receiver deployment and configuration

When the source is exported, place the JAR file in the `<wbic_install>\receiver\was\wbic\userexits` directory on the machine of Company A. You must restart the WebSphere BI Connect receiver component of Company A for it to recognize the modifications.

**Note:** Make sure that the uploaded JAR files are owned by the user `bcguser` and the group `bcggroup`. On AIX, you can use the command:

```
chown bcguser:bcggroup *.jar
```

**Receiver deployment descriptors**

Besides uploading the JAR files, the developer must create a deployment descriptor for the receiver. It is named `ItsoFileMatchingReceiver.xml`, as shown in Figure 18-3 on page 543.

You can find this deployment descriptor in the descriptors folder in the User Exits WebSphere Studio Application Developer project, which you can download from the Web. Refer to Appendix B, “Additional material” on page 773, for details. The schemas used to create the deployment descriptors are in the `\DevelopmentKits\schemas` folder on the WebSphere BI Connect installation CD. They are also included in the downloadable package.

**Adding the receiver to the existing document flow**

The `FileMatchingReceiver` is used by Company A to receive files from the back-end application. This receiver must be imported into the WebSphere BI Connect server of Company A and configured to poll the correct destination.

1. Log in to the Community Console for Company A as the hub administrator. Select **Hub Admin → Hub Configuration → Targets**.
2. In the Target List window (Figure 18-25), a list of currently defined targets is shown. Select **Import Transport Type**.

![Target List](image)

**Figure 18-25**  List of targets on server of Company A
3. In the Import Transport window (Figure 18-26), import the deployment descriptor for the FileMatchingReceiver by providing the file name. For Commit to Database, select Yes. Then click Upload.

Figure 18-26 Importing a new transport
You see the Import was successful message in the Import Transport window (Figure 18-26).

Figure 18-27 New transport imported successfully

To create the target associated to the new transport type, follow these steps:

1. Click Create Target.
2. In the Target Details window (Figure 18-28), complete the following items:
   a. Enter a name for the target, for example FileMatchingRec.
   b. Enter an optional description for the target.
   c. For Transport type, select File Extension Matching from the list. Based on this selection, you see new parameters. These parameters are obtained from the descriptor that was uploaded earlier.
   d. Set Directory to Poll to a valid directory that can be used by bcguser, for example /opt/IBM/WBIConnect/data/companya/FileMatching.
Attention: The user exit uses the `renameTo` method to move the file from the polling directory to a temporary directory within the common storage area (`/opt/IBM/WBICConnect/common`). This `renameTo` method does not support moving files across file systems.

e. For File Extension, Poll Interval, and Thread Number, default values are obtained from the deployment descriptor.

f. Click **Save**.

Figure 18-28  Creating a new target for custom transport
3. When the creation is successful, restart the server component of Company A.

The custom target is now ready to accept new documents for the Company A WebSphere BI Connect instance.

**Attention:** You may need to update the startup script startServer.sh to add another directory to the environment variable WAS_LIBPATH. The /opt/IBM/WBICConnect/receiver/was/wbic/support directory is missing. This folder contains the native library libAIXNative.so. Without adding this directory, you may receive an unsatisfied link error.
18.5.3 Sender configuration

To enable the sender component, similar steps are required. In our scenario, Company E uses the sender user exit. After the JAR file is exported, we place the JAR file in the `<wbic_install>\router\was\wbic\userexits` directory of Company E. Then we restart the Company E router component to recognize the modifications.

Sender deployment descriptors
Besides uploading the JAR files, the developer must create a deployment descriptor for the sender, which is named `ItsoFileMatchingSender.xml` and is shown in Figure 18-15 on page 556.

You can find this deployment descriptor in the folder descriptors in the User Exits WebSphere Studio Application Developer project, which you can download from the Web. Refer to Appendix B, “Additional material” on page 773, for details. The schemas used to create the deployment descriptors are in the `\DevelopmentKits\schemas` folder on the WebSphere BI Connect installation CD. They are also included in the downloadable package.

Adding the Sender to the existing document flow
Company E will use `FileNamingSender` to send from WebSphere BI Connect to the back-end application. This sender exit must be imported into Company E as a new gateway type. A new gateway needs to be created and configured to place files in the correct destination. Finally, the participant connection needs to be updated to use the new gateway.

1. Log in to the Community Console for Company E as the hub administrator.
2. Select **Account Admin → Profiles → Community Participant**.
3. In the Participant Search window (Figure 18-30), click **Search** to display the unfiltered list of participants in the system. Select the profile for **Company E**.

4. Select **Account Admin → Profiles → Gateways**.
5. In the Gateway List window (Figure 18-31), you see the list of gateways defined for Company E. From the list of gateways, select **Import Transport Type**.

---

**Figure 18-31  List of gateways defined for Company E**
6. In the Import Transport Type window (Figure 18-32), complete these tasks:
   
a. Provide the name of deployment descriptor for the new custom sender, that is `ItsoFileMatchingSender.xml`.

b. For Commit to database, select **Yes**.

c. Click **Upload**.

![WebSphere Business Integration Connect Community Console](image)

**Figure 18-32  Importing a new transport type**
7. You see a message when the import was successful. See Figure 18-33. Click **List** to return to the list of defined gateways.

![Figure 18-33   Transport type imported successfully](image)

8. When the list of gateways is shown again, click **Create**.

9. In the Gateway List window (Figure 18-34), complete the following information:
   a. Provide first a name for the gateway, for example **ApplicationFileNamingSender**.
   b. Optionally enter a description.
   c. For Transport, select **File Naming Sender**. This action alters the list of attributes that you can configure. You now see the custom transport attributes that were configured in the deployment descriptor.
   d. Set the Directory Root Path. For example, enter `C:\WBIC\Connect\data\FileNamingSender`.

*Note:* You should have created this directory in advance.
e. For File Extension, select .xml.
f. Click Save.
After saving the new gateway, it is enabled and ready to use as shown in Figure 18-35.

Now that the custom sender and gateway are created, you must associate this new gateway with the existing participant connection between Company A and Company E.

1. Select **Account Admin → Participant Connections**.
2. Select **Company A** as the source and **Company E** as the target and click **Search**.

---

**Figure 18-35  New gateway created**
3. In our environment, two participant connections are shown, but we use only one of them. We use the one that receives AS2 packaged EDI-X12 and sends unpackaged ED-X12, which is the top connection in Figure 18-36. Click Gateways for this connection.
4. In the Edit Participant Connection window (Figure 18-37), for Target Gateways, select **ApplicationFileNamingSender** for production. Click **Save**.

Figure 18-37  Updating the target gateway for the existing connection

5. Restart the router for Company E.

The custom gateway is now ready to send new documents for the Company E WebSphere BI Connect instance.
18.6 Testing the flow

The best approach to testing exits is to isolate them as much as possible. Introducing too much untested code into the system at one time can make troubleshooting much more complex. To isolate each exit, start by configuring WebSphere BI Connect to use a standard flow, taking advantage of built-in receivers and senders. After you test this flow and confirm that it works correctly, replace as few components as possible with custom exits.

During development of the user exits, we always start by configuring the servers to trade a standard EDI-X12 file via AS2. Then, for example, we replace the standard file system receiver with the custom receiver and test it.

The approach used to test the receiver differs from the approach used to test the sender. To test the newly deployed receiver, you must provide it with a file to consume and observe the result.

For example, since the sample solution uses the file system as a transport, we test the custom receiver exit by placing a test file in the file system directory that we configured for the receiver to poll. If the file is consumed by the receiver and passed to the document manager (with the necessary headers set for the connection lookup to succeed), our receiver is functioning correctly. The remainder of the process can then continue as normal. That is, the document manager of Company A packages the EDI-X12 document in AS2 and sends it to Company E.

A sample EDI-X12 file is shown in Figure 18-38. It is no different from the files exchanged in the scenarios described in previous chapters.

Figure 18-38 Sample EDI-X12 document

| ISA*  * * * * companya * companye |
| *040917*1505* *000000005*0*P*: |
| GS*P0* * *20040917*1505*5*004010! |
| ST*850*0005! |
| BEG*00*SA*P345322**20030410! |
| DTM*002*20030410! |
| P01*001*128*EA*98.62! |
| P01*001*100*EA*69.62! |
| SE*6*0005! |
| GE*1*5! |
| IEA*1*000000005! |
Confirm that the receiver is functioning correctly by using these steps:

1. Look at the file system to see if the file is consumed.
2. Check the viewers in the WebSphere BI Connect console for proof that the file has been consumed by the system. Figure 18-39 shows that the AS2 exchange is still successful, even though we now use a custom developed receiver and sender. As before, we see the unpackaged and packaged EDI document and the MDN that is returned by Company E.

![Figure 18-39] Successful AS2 exchange between Company A and E

Testing the sender involves ensuring that the connection, for which the sender is configured as a gateway, is allowed to be executed by supplying WebSphere BI Connect with an appropriate test file. If the sender delivers the file via the transport successfully, that is the file is delivered intact and in the manner intended, the sender has succeeded. For example, in the sample solution, the
sender used the file system as a transport, delivered the file into a specific
directory, and named the file according to configured parameters. You can
confirm this by looking at the file system to see if the file is delivered as shown in
Figure 18-40.

Figure 18-40   View on the file system with XML files created by the sender
For both the sender and receiver and for user exit testing generally, it is good practice to prepare a debugging session, by preparing breakpoints and attaching to the server, prior to allowing the new sender or receiver to be called. Refer to 17.2.9, “Post-deployment tests” on page 519, for details about setting up a debug environment. Figure 18-41 shows a debug session for the custom receiver.

![Debugging the receiver](image)

You can also refer to the system logs for clues as to the source of any problems you encounter. However, the ideal way is to catch them as they happen by interactively stepping through the code.
18.7 Summary

At this point, the WebSphere BI Connect solution between Company A and Company E consists of a custom receiver that polls a directory on the file system for files with a specific and configurable extension (.csv). This document is still an EDI-X12 document. In Chapter 19, “Implementing workflow user exits” on page 587, the file is a comma-separated variable-length file. This EDI-X12 file is processed by the document manager in the same way as before. That is, it is packaged in an AS2 format and sent via the HTTP gateway to Company E.

At Company E, the HTTP target receives the file, and the file is processed by the document manager in the same way as before. That is, the AS2 packaging is removed. The document manager uses the custom sender to write the received file to a configurable directory with a specific extension (.xml). At this time, the file is still an EDI-X12 file, not XML. Chapter 19, “Implementing workflow user exits” on page 587, covers the transformation from CSV to XML.

Figure 18-42 Using the custom receiver and sender in the overall solution
Implementing workflow user exits

This chapter revisits an earlier process flow and relates new concepts to our configuration task. It provides more insight as to how the correct process is invoked based on the business document and participant configurations. Then it relates the flow to our business requirements, and explains how to design and implement a new business exchange.
19.1 Document processing with document workflow

Before we explain user exit development for document processing, we must review a typical document flow in WebSphere BI Connect system and shed some light on the details of the configuration and impact of these settings on the process flow. We use the AS2 configuration used in Chapter 7, “Creating a basic B2B exchange” on page 121, between Company A and Company E to help aid in our discussion.

19.1.1 Component interaction

We use the business requirements from Chapter 7, “Creating a basic B2B exchange” on page 121, to develop an AS2 document flow from Company A to Company E. In doing so, we make several configuration changes.

First we create the targets for the hub. These targets represent the instances of receivers that will be used to facilitate inbound documents. Since targets are configured for the hub, these targets are not directly related to participants but rather represent the inbound options supported by the configured hub.

Next we configure the hub so that the interactions supported by the business processing engine are defined. We create interactions using the console. These configurations represent the document flows that are supported by the hub.

For example, in Chapter 7, “Creating a basic B2B exchange” on page 121, we create an interaction for Company A that enables the hub to process inbound document without a package. The package setting used to represent this is NONE. That is, such packaging as AS2 is not present. Further, the document is received with a protocol identified as EDI-X12 and a Document Flow of ALL.

When a document is received from Company A, going to Company E, and matches the package and protocol criteria, the hub executes the Pass Through action and executes outbound processing. In our example, outbound processing includes wrapping the EDI-X12 message in an AS2 package.

Let us explore these terms more closely.

- **Package**: Packages typically represent industry accepted formats for document exchange. AS2 and RosettaNet are examples of packages that are commonly used today.

- **Protocol**: Protocols typically represent the format of the data wrapped within a given package. These protocols can be electronic data interchange (EDI), XML or other user-defined formats. For example, we use AS2 packaging to wrap a message in the EDI-X12 protocol.
**Document flow** (process): The document flow provides a level of granularity so that different actions can be assigned to specific transaction types. For example, we can represent invoice data using the EDI-X12 format wrapped in AS2 packaging. If we want to process invoice data differently than purchase orders, for example, we can provide this distinction using a document flow specific configuration. The combination of sending and receiving business IDs, a package, protocol and document flow must resolve to one and only one participant connection and hence an action (variable workflow). In some cases, such as in our earlier example, we do not specify this distinction. Instead we specify that ALL documents are processed using the same action (Pass Through).

**Action**: Actions are associated to variable workflow. Actions represent a unique series of steps or execution blocks. These steps can be arranged in any given sequence, hence the term *variable* workflow.

Now that we have created an interaction, we can associate this interaction to the participants in the system. We created the participants that represent Companies A and E and set the appropriate attributes. We use such attributes as a business identifier to uniquely identify participants in the system.

In addition to these basic identifiers, we must configure a participant's B2B capabilities. B2B capabilities are used to identify all of the forms of integration that are possible for a given participant. We use the B2B capabilities view to enable and disable the forms of interactions that are possible for a particular participant. For example, we enable Company A so that it can receive inbound business documents with NONE/EDI-X12 packaging and protocol. We also enable Company A so that it can send business documents with AS2/EDI-X12 packaging and protocol.

Next, we create gateways for these new participants. *Gateways* are configured instances of senders and represent the exit points of the WebSphere BI Connect system. Unlike targets, gateways are associated to participants.

Finally, in Chapter 7, “Creating a basic B2B exchange” on page 121, we configure the system so that Company A and Company E use this interaction to exchange documents. This configuration, called a *participant connection*, drives the events that transpire as new documents are received by the hub and match the package, protocol, and document flow settings.

By configuring the attributes and activating the participant connection, we complete the necessary steps to represent a B2B document exchange between Company A and Company E. Let us look at how the WebSphere BI Connect business processing engine uses this configuration.
19.1.2 Document processing

Recall from our discussion about receivers, as a new document is received by the WebSphere BI Connect system, many headers are set. The receiver is required to set some of these headers so WebSphere BI Connect can move to the next step of processing.

- **BCG_REQUEST_URI**: The URI of the receiver, for example the path of the directory the receiver is inspecting or the URL of the receiver for HTTP
- **RCVR_DESTINATION**: The destination partner for this connection
- **BCG_RCVD_MSG_LENGTH_INC_HDRS**: The length of the message including headers
- **BCG_RCVR_DOC_TIMESTAMP**: A timestamp for when the document was received

In fact, there are multiple check points where BCG headers can be set in the WebSphere BI Connect document process. The correct document flow or participant connection must be identified by the business process engine using these BCG headers. In any given document flow, the steps of the inbound fixed workflow are responsible for setting the additional headers required to select the appropriate variable workflow or action. As such, you must set several BCG headers at a minimum before the inbound fixed workflow is complete.

- **BCG_FRPACKAGINGCD**: The name of the package in which the document is enclosed at the start of this connection
- **BCG_FRPACKAGINGVER**: The version of the package in which the document is enclosed at the start of this connection
- **BCG_PKG_INITBUSINESSID**: The business ID of the partner who started this connection, which may not be the same as from whom the document came last
- **BCG_PKG_FRBUSINESSID**: The business ID of whom the document came from last, at a package level
- **BCG_PKG_TOBUSINESSID**: The business ID of whom the document is destined for, at a package level
- **BCG_INITBUSINESSID**: The business ID of the partner who started this connection, which may be different than the package value
- **BCG_FRBUSINESSID**: The business ID of whom the document came from last, which may be different than the package value
- **BCG_TOBUSINESSID**: The business ID of who the document is destined for, which may be different than the package value
- **BCG_FRPROTOCOLCD**: The name of the protocol for the document
The combination of these BCG headers identifies the correct participant connection and ultimately resolves to an action or variable workflow.

After the execution of the configured action, the outbound processing requirements are read from the participant connection’s packaging and protocol settings. The WebSphere BI Connect system executes the necessary outbound fixed workflow steps. In the case of our example in Chapter 7, “Creating a basic B2B exchange” on page 121, the outbound message from Company A is wrapped in AS2 packaging.

Finally, the configuration of the participant connection identifies the gateway for the destination. The source gateway indicates the gateway used by WebSphere BI Connect in case of a synchronous request. The target gateway identifies the gateway to be used by WebSphere BI Connect to send the request to the destination (Company E).

The document is then passed to the gateway, so the gateway can interact with the transport. In the case of our example, the document is sent to Company E using the HTTP gateway. When a message is received by Company E, the document resolution process is repeated given Company E’s interactions and participant connection configuration.

Note: If these headers are not set properly, a channel resolution error is raised by the business process engine.
19.2 Business case and document processing requirements

In our ITSO business case, a few requirements must be addressed by our custom workflow components.

- **Custom package**: A custom package is required. The custom packaging was defined by the ITSO organization. The extensions in Company A have to package the outbound document in the ITSO format. Conversely, Company E’s inbound document process must unpackage the business document.

- **Encryption and decryption**: Security is required and the extensions have to support encryption and decryption using the WebSphere BI Connect provided security libraries. Company A has to encrypt the document before sending the information via the configured HTTP gateway. Company E must decrypt the inbound document before the business data can be processed.

- **CSV to XML**: Company A needs to transform the CSV documents produced by the Company A back-end application to the required XML schema required by the ITSO organization. For our example, ITSO provides an invoice XML schema that is used to exchange invoice data between Company A and Company E in the custom ITSO packaging.

Given these requirements, we summarize the custom development that is required for Company A and Company E as shown here and in Figure 19-1.

- **Company A**
  - Inbound Fixed Workflow
    - Read routing and document type information found in the inbound CSV file from Company A’s application.
  - Variable Workflow
    - Transform the CSV-based document from the application to the required ITSO XML schema for invoices.
  - Outbound Fixed Workflow
    - Encrypt the outbound business document.
    - Package the outbound business document.

- **Company E**
  - Inbound Fixed Workflow
    - Decrypt the inbound business document.
    - Unpackage the ITSO specific wrapper.
19.2.1 ITSO specific requirements

The ITSO organization has imposed two specific requirements on the business case. The ITSO organization defines:

- The format (XML schema) of the business document exchanged between participants
- The format of the package wrapper around the business data that is exchanged

ITSO business documents

In our use case, Company A and Company E exchange invoice data. The invoices originate from Company A and are sent to Company E. You can find the schema (ItsoInvoice.xsd) in the “schemas” directory in the project files included and shown in Figure 19-2.
Figure 19-2  XML schema used for creating ITSO business documents

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="unqualified" attributeFormDefault="unqualified">
    <element name="ITSO_INVOICE">
        <complexType>
            <sequence>
                <element name="protocolName" type="string"></element>
                <element name="protocolVersion" type="string"></element>
                <element name="Version" type="string"></element>
                <element name="From" type="string"></element>
                <element name="To" type="string"></element>
                <element name="AccountNumber" type="string"></element>
                <element name="OrderNumber" type="string"></element>
                <element name="Shipping">
                    <complexType>
                        <sequence>
                            <element name="ContactName" type="string"></element>
                            <element name="ContactPhone" type="string"></element>
                            <element name="ShippingAddress" type="string"></element>
                        </sequence>
                    </complexType>
                </element>
                <element name="Items">
                    <complexType>
                        <sequence>
                            <element name="Item" minOccurs="1" maxOccurs="unbounded">
                                <complexType>
                                    <sequence>
                                        <element name="Number" type="string"></element>
                                        <element name="Description" type="string"></element>
                                        <element name="Quantity" type="string"></element>
                                        <element name="UnitPrice" type="string"></element>
                                        <element name="SubTotal" type="string"></element>
                                    </sequence>
                                </complexType>
                            </element>
                        </sequence>
                        <attribute name="numItems" type="integer"></attribute>
                    </complexType>
                </element>
                <element name="OrderTotal" type="string"></element>
            </sequence>
        </complexType>
    </element>
</schema>
```
**ITSO package**

The ITSO package wraps the aforementioned business document in the ITSO format. The custom package uses CSVs to represent the wrapper information. The following values are included in the wrapper:

- **PackageId**: Defines the type of package
- **Version**: Defines the version of the package
- **From**: Defines the trading partner that is the source of the business document
- **To**: Defines the trading partner that is the destination of the business document
- **Payload**: Contains a business document in XML that meets the requirements of the ITSO schema

Figure 19-3 shows a sample message sent from Company A and Company E including the business data and ITSO specific wrapper.
Figure 19-3  Sample business document and package
19.2.2 Business object framework

When developing our custom extensions, the team established a model that isolates the complexity of our packaging and transformation requirements and creates a framework to facilitate the construction of business documents. The business object framework provides the following benefits:

- Isolates business logic for marshalling and unmarshalling business documents
- Adds the ability to test the main transformation and marshalling logic outside of the scope of the WebSphere BI Connect environment
- Simplifies the implementation of the workflow extensions

This framework is provided with the source code included in the project and can be manipulated to meet your specific project requirements. The bulk of the business framework logic is contained in the com.ibm.itso package. JUnit classes used to test the business object framework are contained in the com.ibm.itso.test package.

This business object framework is not the focus of this redbook. However, it is worth discussing briefly how the user exits leverage the logic provided by the framework classes.

As part of the sample business case, business documents, such as invoices, are sent, received, and manipulated by the participants. Rather than coupling this business logic to the user exit classes, we create a business object framework to encapsulate the data and methods required. This has the additional benefit of allowing us to test most of the business logic (and consequently most of the overall logic) separately from the WebSphere BI Connect server, reducing the time necessary to develop, deploy, and test considerably.

An abstract BusinessObject class is designed to encapsulate the common data elements and provide abstract methods for the most common functions that can be overridden by more specific classes. Business document types, such as Invoice and ITSOPackage, are at the next level of the hierarchy. These classes hold the common data elements for the type of document they represented. This level incorporates the logic for copying from one business object (of the same document type) to another. At the most specific level of the hierarchy are business documents of a particular format such as XmlInvoice. These classes cover the functions necessary to marshall and unmarshall the documents from and to the specific format.

In addition to the business document classes, a factory pattern is used and a corresponding factory class, BusinessObjectFactory, is created. This allows the client application to request the type of business object required including the document type and format.
By constructing the business object framework in this way, it is possible to work with business objects generically. For example, a business object is instantiated via the BusinessObjectFactory:

\[
bo = \text{bof.getBusinessObject(process, protocol)};
\]

Here, \textit{process} is the document type, and \textit{protocol} is the format, for consistency with WebSphere BI Connect terminology.

Typically the two pieces of information necessary to instantiate the correct business document are read at run time by passing the incoming file. This makes sense since it allows the user exit to deal with any number of business document and format combinations by constructing corresponding classes. To allow document types and formats to be mapped to different class names, a configuration file is used, \text{businessobjectfactory.config}, which consists of name/value pairs of the pattern \text{<document type>-<format>=<classname>}. The configuration file for our business case is shown in Figure 19-4.

```bash
# Business Object Support Configuration
# process-format=classname
ITSO_INVOICE-ITSO_XML=com.ibm.itso.XmlInvoice
ITSO_INVOICE-COMPA_CSV=com.ibm.itso.CsvInvoice
ITSO-CSV=com.ibm.itso.CsvItsoPackage
```

\textit{Figure 19-4  Configuration file for the business object framework}

As discussed, the most specific classes, such as XmlInvoice, hold the necessary logic to marshall and unmarshall the documents from files to objects and back. In the case of XML formatting, an open source Java XML framework called \textit{JaxMe} was used. JaxMe is built on top of JAXB. JaxMe classes are generated from the XML schema by creating an ANT build file that specifies the name of the schema and the path to generate the XML manipulation classes into. We use these generated JaxMe classes to handle the marshalling and unmarshalling of our business documents from and to XML.

As best practice given the benefits described here, consider using a business object framework pattern or similar method.
19.2.3 Business case development

In a common customer development scenario, it is likely that Company A and Company E have separate development teams and code the extension specific to their own unique perspective. That is, Company E does not require CSV to XML transformation. However, Company E requires the decryption logic.

For simplicity, the development team creates all of the extensions needed to support Company A and Company E’s business requirements in one project. The resulting JAR file derived from the project is exported and distributed to both Company A and Company E’s WebSphere BI Connect instances.

19.3 Designing the document process

Before you can create the code for the document process, you must determine the nature of the process flow and clearly articulate the values of the:

- Package
- Protocol
- Document Process

This is required because the resolution process described earlier uses these values (literally) to determine the appropriate participant connection. You must develop the custom inbound fixed workflow components with this information defined, to set the aforementioned BCG headers with the appropriate values to facilitate business process resolution.

19.3.1 Creating document flow definitions

This section describes the hub configuration necessary to create the required inbound and outbound document flow settings for both Company A and Company E.

Company A
Company A requires document flow settings to describe the inbound and outbound package, protocol, and document flow relationship.

Inbound document flow
The design team creates a document flow definition that describes our inbound document flow from Company A’s application. The application sends WebSphere BI Connect an unpackaged document that contains the proprietary delimited format of Company A. The information contained in the delimited format is related to the ITSO invoice defined by the ITSO organization. Given these requirements, we derive the parameters and values listed in Table 19-1.
The value COMPA_CSV is defined by the development team. This can be any value, but we create the literal COMPA_CSV to describe the nature of the proprietary protocol of Company A.

We also choose ALL for the document flow because we are designing the workflow components so that these components can be reused for any transaction type (Invoice, Purchase Orders, and so on). We anticipate that Company A and Company E want to exchange multiple transaction types.

**Outbound document flow**

ITSO insists that the outbound document from Company A packaging and protocol comply with the organization’s requirements. To meet these requirements, the design team created a document flow that reflects the parameters and values listed in Table 19-2.

*Table 19-2  Document flow parameters and values*

<table>
<thead>
<tr>
<th>Package</th>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ITSO</td>
<td>1.0</td>
</tr>
<tr>
<td>Protocol</td>
<td>ITSO_XML</td>
<td>1.0</td>
</tr>
<tr>
<td>Document Flow</td>
<td>ALL</td>
<td>ALL</td>
</tr>
</tbody>
</table>
Creating document flow definitions for Company A

We use the Console to create the document flow definitions for Company A.

1. Log in to Company A’s WebSphere BI Connect instance as the hub administrator. Select Hub Admin → Hub Configuration → Document Flow Definitions.

2. Figure 19-5 shows the document flow definitions of the base configuration developed in Chapter 7, “Creating a basic B2B exchange” on page 121. Click Create Document Flow Definition.

Figure 19-5  Document flow definitions

3. In the Create Document Flow Definitions window (Figure 19-6), enter the protocol information for the CSV format of Company A:
   a. For Document flow type, select Protocol.
   b. For the Code and Name parameters, enter the value COMPA_CSV.
   c. For Version, type 1.0.
   d. Type a description.
   e. For Visibility, select the appropriate option.
   f. Select Package: None as shown in Figure 19-6, so that COMPA_CSV is included in this package.
   g. Click Save.
Figure 19-6  Creating the COMPA_CSV protocol
4. Repeat the process explained in step 3 for the ITSO package. Enter the ITSO package information as shown in Figure 19-7. Set the appropriate visibility and select Top Level to indicate that the package ITSO configuration represents a top level package. Click Save.
5. Repeat the process explained in step 3 one more time for the ITSO_XML protocol. Indicate that the ITSO_XML protocol belongs to the ITSO package, as shown in Figure 19-8. Click Save.

*Figure 19-8 Creating the ITSO_XML protocol*
Figure 19-9 shows the list of document flow definitions, expanded so that the new document flows are visible. The COMPA_CSV protocol is placed under the None package. There is a new package and protocol entry for ITSO and ITSO_XML.
Now associate ALL to the document flow level of the configuration. The ALL entry exists and new protocols can be associated to the document flow level by selecting the magnifying glass icon next to an existing entry.

1. You can find an existing entry under the None/EDI-X12 package/protocol combination as shown in Figure 19-9. When you see the detailed view, click Add New Context.
2. Set the configuration as shown in Figure 19-11. Associate the document flow with the protocols COMPA_CSV and ITSO_XML and click **Save**.

![Update Document Flow Definitions](image)

**Figure 19-11  Adding new context for a document flow**
3. Return to the list of document flows and expand the changed and new levels. You should see that the document flow ALL is now associated with the right protocols, as shown in Figure 19-12.

![Figure 19-12 Updated document flow definitions](image)

**Company E**

Company E requires similar document flow definitions to describe the inbound and outbound package, protocol, and document flow relationship.

**Inbound document flow**

We created a document flow definition that describes the inbound document flow from Company A. Company A sends Company E an ITSO packaged document that contains an ITSO XML document. The transaction described by the XML document is the ITSO invoice defined by the ITSO organization. Given these requirements, the design team creates a document flow that reflects the parameters and values in Table 19-3.
Table 19-3  Document flow parameters and values

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>ITSO</td>
</tr>
<tr>
<td>Protocol</td>
<td>ITSO_XML</td>
</tr>
<tr>
<td>Document Flow</td>
<td>ALL</td>
</tr>
</tbody>
</table>

Again, we choose ALL for the document flow because we are designing the workflow components so that these components can be reused for any transaction type (Invoice, Purchase Orders, and so on).

**Outbound document flow**

Outbound from WebSphere BI Connect, we must create a document that is ready for Company E’s back-end applications. To meet this need, our design team creates a document flow that reflects the parameters and values in Table 19-4.

Table 19-4  Document flow parameters and values

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>None</td>
</tr>
<tr>
<td>Protocol</td>
<td>ITSO_XML</td>
</tr>
<tr>
<td>Document Flow</td>
<td>ALL</td>
</tr>
</tbody>
</table>
Creating Company E's document flow definitions

Using the Console to create the document flow definitions for Company E, follow these steps.


2. In the Manage Document Flow Definitions window (Figure 19-13), you see the document flow definitions that were used in previous chapters. Click Create Document Flow Definition.

3. In the Create Document Flow Definitions window (Figure 19-14), enter the values for the ITSO package outlined in Table 19-3. Set the appropriate
visibility and select the **Top Level** option at the bottom of the page to indicate that this is a top level protocol. Click **Save**.

![Image of WebSphere Business Integration Connect Community Console](image)

**Figure 19-14   Creating the ITSO package**

4. Repeat the same process to create a new document flow definition for the ITSO.XML protocol outlined in Table 19-4 on page 609.
As shown in Figure 19-15, set the appropriate visibility. This time, select the packages that you want to associate with the ITSO_XML protocol. For our scenario, we associate this protocol with two packages:

- **ITSO** for the inbound flow from Company A
- **None** for the outbound flow to the back-end application of Company E

*Figure 19-15   Creating the ITSO_XML protocol*
5. Add the document flow ALL for both packages. Return to the list of document flow definitions and locate the document flow for EDI-X12 by expanding the list of definitions. Click the magnifying glass icon for this document flow. When the contexts of this document flow is shown, click **Add New Context**.

---

**Figure 19-16  List of document flow definition contexts**
6. As shown in Figure 19-17, select the check boxes for the **ITSO_XML** protocol for the **None** and **ITSO** packages. Click **Save**.

![Update Document Flow Definitions](image)

**Figure 19-17 Adding new context**

7. As shown in Figure 19-18, return to the list of document flow definitions and expand them, so that you can verify that document flows, protocols and packages are added where expected.
**Note:** Contrast the modeling techniques between customXML/purchaseOrder and ITSO_XML/ALL. If Company E needs a new custom XML format, add a new document flow to the protocol customXML. For the ITSO protocol, handling an additional document does not require adding a new document flow. At the document flow level, no specific processing is required for different types of documents within the ITSO protocol. For the customXML protocol, you can easily associate different processing with each business document since you modeled them separately.
### 19.3.2 Creating an interaction

After the package, protocol, and document flow options are introduced to the system, you can create interactions that define valid combinations of document processes. In our scenario, we create an interaction in each Company’s hub.

#### Company A

Company A must support an interaction that processes inbound None/COMPA_CSV/All documents and returns ITSO/ITSO_XML/All documents. To create this interaction, follow these steps:

1. Log into Company A as the hub administrator. Select **Hub Admin → Hub Configuration → Document Flow Definition**.

2. Click **Manage Interactions**.

3. A window opens that allows you to search for interactions matching certain criteria (see Figure 19-19). Click **Create Interaction**.

4. In the Manage Interactions window (Figure 19-20), complete these tasks:
   a. Expand the source and target document flows to locate None/COMPA_CSV/ALL as the source and ITSO/ITSO_XML/ALL as the target document flow. Select the radio button for the source and target document flows.
b. For Action, select **Pass Through**.

**Note:** We temporarily set this action to Pass Through. When we create the custom transformation action for Company A, we change this action to specify our custom ITSO transformation.

c. Click **Save**. We now have an interaction that receives None/COMPA_CVS/ALL documents and returns ITSO/ITSO_XML/ALL documents.
5. When you return to the view Manage Interactions, click **Search**. You should see the new interaction between ones that were created earlier during the implementation of our scenario. See Figure 19-21.

![Manage Interactions](image)

**Figure 19-21** List of interactions

Later we use this interaction to create a participant connection between Company A and Company E.
Company E
Company E must create an interaction that processes ITSO/ITSO_XML/ALL documents and returns None/ITSO_XML/ALL documents. To create this interaction, complete these steps:

1. Log into Company E as the hub administrator. Select Hub Admin → Hub Configuration → Document Flow Definition.
2. Click Manage Interactions.
3. When the list of interactions is shown, click Create Interaction.
4. In the Manage Interaction window (Figure 19-22), complete these tasks:
   a. Expand the source and target document flows to locate ITSO/ITSO_XML/ALL as the source and None/ITSO_XML/ALL as the target document flow. Select the radio button for the source and target document flows.
   b. For Action, select Pass Through.
   c. Click Save.
Figure 19-22  Creating a new interaction: Company E
5. Return to the Manage Interactions window. This time, as an example, perform a search using the drop-down lists. Review that the new interaction is listed as expected. See Figure 19-23.

![Manage Interactions](image)

**Figure 19-23  Filtered list of interactions**

We now have an interaction that receives ITSO/ITSO_XML/ALL documents and returns None/ITSO_XML/ALL documents. Later, we use this interaction to create a participant connection between Company A and Company E.
19.3.3 Configuring B2B capabilities

For a participant to use our interaction, we must set the B2B capabilities for the participants involvement in the document flow. In our use case, Company A accepts COMPA_CSV files from the back-end application and sends ITSO data to Company E. Company E accepts ITSO information and sends the documents to Company E’s back-end application after removing the ITSO package.

Each WebSphere BI Connect instance needs to be configured appropriately to represent these capabilities within the profiles of the participants.

Company A
B2B capabilities are associated with the participants who are configured within the WebSphere BI Connect instance. The Company A instance must be configured specific to Company A’s perspective of the document flow.

1. If you don’t already have an active session, log into Company A as the hub administrator. Select Account Admin → Profiles → Community Participant.

2. In the Participant Search window (Figure 19-24), click Search. Select Company A and navigate to Company A’s B2B capabilities.
Figure 19-24  Participant list
3. You see the B2B Capabilities window (Figure 19-25) for Company A. Company A is capable of being a source of COMPA_CSV and ITSO_XML documents.

![WebSphere Business Integration Connect Community Console](image)

Figure 19-25 B2B capabilities of Company A on the server of Company A
Enable Company A as shown in Figure 19-26.

![Profile > Company A > B2B Capabilities](image)

**Figure 19-26  Updated B2B capabilities of Company A on the server of Company A**

**Attention:** Make sure that you work with the profile of Company A, and not the Operator. Review the heading under the menu bar. It should include the name of the profile with which you work.

4. Return to the list of participants (see Figure 19-24 on page 623). This time, select the profile of **Company E**.
5. Company E is capable of being a target for ITSO.XML documents. Configure the Company E participant as shown in Figure 19-27.
Company E

B2B capabilities are associated with the participants configured within the WebSphere BI Connect instance. The Company E instance must be configured specific to Company E’s perspective of the document flow.

1. Log into Company E as the hub administrator. Select Account Admin → Profiles → Community Participant.

2. In the Participant Search window (Figure 19-28), click Search.


![Figure 19-28   Participant list](image)
4. Company A is capable of being a source of ITSO documents. Enable Company A as shown in Figure 19-29.

Figure 19-29   Updated B2B capabilities of Company A on the server of Company E
5. Company E is capable of being a target for ITSO documents. Configure the Company E participant as shown in Figure 19-30.

![WebSphere Business Integration Connect Community Console](image)

**Figure 19-30** Updated B2B capabilities of Company E on the server of Company E

**Note:** We enabled too many B2B capabilities. For each profile on each server, we enabled the source and target flow. However, we could have set the B2B capabilities more strictly. For example, in Figure 19-30, it is sufficient to enable the target option for None/ITSO_XML/ALL. The source option is redundant in this case.

You have now completed the configuration of the two WebSphere BI Connect instances to support ITSO packaging and protocol ITSO_XML. However, at this time, the servers do not know how to build an ITSO package or how to remove it. The following sections discuss the supporting code.
19.4 Developing a fixed workflow

A fixed workflow has a defined sequence of steps that you can use for any particular document flow. There are two types of fixed workflows:

- Inbound
- Outbound

Inbound and outbound fixed workflows are independent from one another, but their functions are typically developed complimentary. Typical features implemented in a fixed workflow are:

- Compress or decompress documents
- Encrypt or decrypt documents

WebSphere BI Connect provides a number of fixed workflow capabilities with the base product. For example, as you saw in the previous example, WebSphere BI Connect provides the ability to wrap and unwrap AS2 packages. Refer to the WebSphere BI Connect documentation for a complete list of provided fixed workflows.

To extend these given capabilities, WebSphere BI Connect provides a BusinessProcessHandlerInterface that a developer can implement to create new fixed workflow steps. Both types of fixed workflow (inbound and outbound) can be extended using this interface.

19.4.1 BusinessProcessHandlerInterface

The developer must implement this interface when creating new fixed workflow steps. These steps are then introduced to the WebSphere BI Connect system and invoked using the handler chain concept previously discussed. The main methods of this interface are: applies(), init(), and process().

applies()
The business process engine invokes the applies() method when a new business document is received. If the fixed workflow step can process the inbound request, the developer returns the boolean value of true.

init() The init() method is called by the framework to initialize variables after the applies() method returns true.

process() The process() method is called by the business process engine after the init() method. The process() method should contain the logic necessary to handle the
request. For example, the process() method for an inbound fixed workflow should set the necessary BCG headers to facilitate channel resolution. A fixed outbound workflow may use the process() method for document packaging and encryption.

19.4.2 Fixed inbound workflow

The business processing engine runs a fixed inbound workflow as new documents are introduced to the system. The steps of the inbound fixed workflow are configured in WebSphere BI Connect using handler chains. A handler chain is available for each inbound fixed workflow step. There are two steps in an inbound fixed workflow:

1. Protocol parse
2. Protocol package

19.4.3 Fixed outbound workflow

The business processing engine runs the fixed outbound workflow as the business document is received from the variable workflow engine. There is only one outbound user exit step. This exit can be used to execute any task before the document is passed to the sender/gateway. This exit is commonly used for document encryption, document compression and so on.

The outbound fixed workflow handler chain is configured using the Console in the same way the inbound fixed workflow is configured.

19.5 Developing a variable workflow

Variable workflow is defined using actions. Developers may create new actions or copy existing actions.

19.5.1 BusinessProcessFactoryInterface

BusinessProcessFactoryInterface represents the standard interface for all variable workflow. The WebSphere BI Connect system uses a factory pattern for creating variable workflow. This pattern allows the WebSphere BI Connect system to control the type and number of instantiated workflows. The main methods of this interface are getBusinessProcess() and returnBusinessProcess().
**getBusinessProcess()**  
The WebSphere BI Connect system calls the `getBusinessProcess()` method on the factory. This method returns an instance of a class that implements the `BusinessProcessInterface` class. The `getBusinessProcess()` method helps to pass in the configuration in the form of a Config object and the `BusinessDocumentInterface` for processing.

**returnBusinessProcess()**  
The `returnBusinessProcess()` method is called by the WebSphere BI Connect system when processing completes.

### 19.5.2 BusinessProcessInterface

BusinessProcessInterface is exposed to execute processing logic. Classes that implement this interface encapsulate the logic for the action. The main methods are `process()` and `reset()`.

**process()**  
The `process()` method is the main method for the class. This method is called by the WebSphere BI Connect system to execute the necessary logic on the business document.

**reset()**  
The `reset()` method is provided to the developer to clean up resources before the process is terminated. The `reset()` method is called by the `BusinessProcessFactory`.

### 19.6 Creating the ITSO workflow

To give a logical flow to our discussion, we address each component required in our document processing solution in the order by which they are called as the document moves from Company A to Company E.

#### 19.6.1 Company A

The following sections discuss the development effort necessary for Company A to meet the stated requirements.
Inbound fixed workflow requirements and design
In addition to the ITSO wrapping and unwrapping, Company A has a requirement to create a fixed inbound workflow that is capable of reading the CSV file from the Company A application and extracting the routing information and process type.

CompanyAInboundProcessor
The CompanyAInboundProcessor is the first document processing extension called in our scenario. This protocol parsing handler is responsible for:

- Marshalling the inbound CSV file into the business object framework
- Setting the BCG header to identify the appropriate participant connection. In this case, the invoice document is received from the Company A application and goes to Company E.

The role of the main applies(), process() and init() methods is explained here:

- **applies()**: The method is called by the framework to determine if the CompanyAInboundProcess can manage the processing of the file. In this applies() method, we use the file contents to determine if this document is an ITSO document that requires conversion. The inbound CSV document is marshalled into our business object framework.
  
The business object is then stored in the BusinessDocument as a temporary object so it may be used by other WebSphere BI Connect components. See Figure 19-31.
public boolean applies(BusinessDocumentInterface busDoc) {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
            this.getClass().getName(),
            "Entering applies()");

    boolean applies = false;
    File doc = busDoc.getDocument();
    BusinessObjectFactory bof = null;
    try {
        bof = BusinessObjectFactory.getInstance();

        String protocol = null, protocolVersion = null,
                            process = null, processVersion = null;

        BusinessObjectUtil bou = new BusinessObjectUtil();
        Object[] docDetails = bou.getCsvElements(doc,3);
        protocol = (String)docDetails[0];
        process = (String)docDetails[2];

        bo = bof.getBusinessObject(process,protocol);
        bo.unmarshall(doc);

        if((bo.getProtocolName().equals(ExitConstants.COMPANYA_PROTOCOL_CODE)
                &
                bo.getProtocolVersion().equals(ExitConstants.COMPANYA_PROTOCOL_VERSION)) {
            applies = true;
        }
        busDoc.setTempObject(ExitConstants.TMP_BUSINESS_OBJ,bo);
    } catch (Exception e) {
        applies = false;

        bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
                this.getClass().getName(),
                e.getClass().getName() + ": " + e.getMessage() + ": " +
                e.toString());
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
            this.getClass().getName(),
            "Leaving applies()");
    return applies;
}
**init()**: This method remains blank because our design does not require any specific initialization code.

**process()**: The `process()` method (Figure 19-32) must set the required BCG headers that lead to document conversion. By setting these headers, `CompanyAInboundProcessor` helps the processing engine resolve to the correct participant connection.

These settings are related to the configuration that we made for the Company A interaction. As stated, the BCG headers that are set in the method are used to resolve to a participant connection.

```java
// job here is to set the necessary bcg headers
if (bo == null) {
    params = new String[1];
    params[0] = "BusinessObject was null";
    ei = new EventInfo("BCG240615", busDoc, params);
    eiArr = new EventInfo[1];
    eiArr[0] = ei;
    busDoc.addEvents(eiArr);
    busDoc.setDocumentState(BCGDocumentConstants.BCG_DOCSTATE_FAILED);
} else {
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPACKAGINGCD, ExitConstants.NONE_PKG);
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_FRBUSINESSID, bo.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_INITBUSINESSID, bo.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_TOBUSINESSID, bo.getTo());
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRBUSINESSID, bo.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_TOBUSINESSID, bo.getTo());
    busDoc.setAttribute(BCGDocumentConstants.BCG_INITBUSINESSID, bo.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROTOCOLCD, ExitConstants.COMPANYA_PROTOCOL_CODE);
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROTOCOLVER, ExitConstants.COMPANYA_PROTOCOL_VERSION);
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROCESSCD, ExitConstants.ALL_VERSION);
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROCESSVER, ExitConstants.ALL_VERSION);
}
```

*Figure 19-32  The process() method*
Variable workflow requirements and design
The ITSO business requirements dictates that the invoice data received from the Company A application is converted to the associated Invoice XML schema defined by ITSO.

The design team has developed a variable workflow to carry out this transformation. We use the business object framework to abstract the details of the transformation.

TransformationFactory
The TransformationFactory class (Figure 19-33) implements the BusinessProcessFactoryInterface. This class is used to distribute processing instances on and off the stack. When the factory class is instantiated, it caches instances of the ItsoTransformer class.

```java
public TransformationFactory() {
    bcgUtility = new BCGUtil();
    // example of caching
    transformers = new Stack();
    
    cacheSize = 4; // in production would be read in from a property file
    for (int i = 0; i < cacheSize; i++) {
        transformers.push(new ItsoTransformer());
    }
}
```

Figure 19-33  TransformationFactory
**getBusinessProcess()**: This method (Figure 19-34) returns an instance of the ItsoTransformer class. The ItsoTransformer class contains the processing logic for the transformation.

```java
public BusinessProcessInterface getBusinessProcess(
    Context context,
    Config config,
    BusinessDocumentInterface busDoc) {

    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Entering getBusinessProcess()");

    BusinessProcessInterface bpi = null;
    if(transformers.isEmpty()) {
        bpi = new ItsoTransformer();
    } else {
        bpi = (BusinessProcessInterface)transformers.pop();
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Leaving getBusinessProcess()");
    return bpi;
}
```

*Figure 19-34  The getBusinessProcess() method*

**returnBusinessProcess()**: This method (Figure 19-35) places the instance of the ItsoTransformer back in the cache.

```java
public void returnBusinessProcess(BusinessProcessInterface busProcess) {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Entering returnBusinessProcess()");

    transformers.push(busProcess);

    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Leaving returnBusinessProcess()");
}
```

*Figure 19-35  The returnBusinessProcess() method*
**ItsoTransformer**

The ItsoTransformer interacts with the business object framework to transform the CSV data to the required ITSO XML structure.

- **process()**: The `process()` method (Figure 19-36) instantiates two instances of the business object class that represent the CSV and XML formats. These Java classes are used to copy CSV data to the XML equivalent.

```java
bo1 = (BusinessObject)busDoc.getTempObject(ExitConstants.TMP_BUSINESS_OBJ);
if(bo1 == null){
    params = new String[1];
    params[0] = "BusinessObject was null";
    //Raise Event
    busDoc.setDocumentState(BCGDocumentConstants.BCG_DOCSTATE_FAILED);
} else {
    String process = bo1.getName();
    String toProtocolName = (String)busDoc.getAttribute(BCGDocumentConstants.BCG_TOPROTOCOLCD);
    String toProtocolVer = (String)busDoc.getAttribute(BCGDocumentConstants.BCG_TOPROTOCOLVER);
    bof = BusinessObjectFactory.getInstance();
    BusinessObject bo2 = bof.getBusinessObject(process,toProtocolName);
    bo1.copyInto(bo2);
    bo2.setProtocolName(toProtocolName);
    bo2.setProtocolVersion(toProtocolVer);
    File xmlFile = busDoc.createFile();
    bo2.marshall(xmlFile);
    busDoc.setTempObject(ExitConstants.TMP_BUSINESS_OBJ,bo2);
    busDoc.setDocument(xmlFile);
}
```

*Figure 19-36  The process() method*

- **reset()**: This method remains blank since our design did not require any specific logic to reset the class.

**Outbound fixed workflow requirements and design**

For our ITSO business case, we need to create custom outbound fixed workflow to support the Company A to Company E interaction. The outbound workflow must encrypt the ITSO invoice and wrap the business document in the custom ITSO package. To meet this requirement, the design team created the ItsoCustomFWPackager that implements BusinessProcessHandlerInterface.
ItsoFWPackager
The main methods of this class are:

- **applies()**: The applies() method (Figure 19-37) checks values from the “To” protocol name and version. If these values match the configured document flow definition values of ITSO/1.0, the applies() method returns true.

```java
public boolean applies(BusinessDocumentInterface busDoc) {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Entering applies()");
    boolean applies = false;
    if(((String)busDoc.getAttribute(BCGDocumentConstants.BCG_TOPROTOCOLCD)).equals(ExitConstants.ITSO_XML_PROTOCOL_CODE) &&
    ((String)busDoc.getAttribute(BCGDocumentConstants.BCG_TOPROTOCOLVER)).equals(ExitConstants.ITSO_XML_PROTOCOL_VERSION)) {
        applies = true;
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Leaving applies()");
    return applies;
}
```

*Figure 19-37  The applies() method*

- **init()**: This method remains blank since our design did not require any specific initialization code.

- **process()**: The process() method (Figure 19-38 and Figure 19-39) is responsible for encrypting the document and wrapping the results in an ITSO package. This method uses the security features that are provided by the WebSphere BI Connect system.
Figure 19-38  The process() method (Part 1 of 2)
19.6.2 Company E

The following sections discuss the development effort necessary for Company E to meet the stated requirements.

**Inbound fixed workflow requirements and design**

Company E needs to create extensions to their inbound fixed workflow to meet the following requirements:

- Unwrap the custom ITSO package
- Decrypt the inbound business document

To do so, the design team creates two classes that implement the BusinessProcessHandlerInterface: ItsoFWUnpackager and ItsoInboundFWProcessor.

**ItsoFWUnpackager**

The ITSOFWUnpackager is responsible for unwrapping the custom ITSO package. Its main methods are:
- **applies()**: This method (Figure 19-40) returns *true* if the document is packaged in the ITSO package and therefore can be handled by this class.

```java
public boolean applies(BusinessDocumentInterface busDoc) {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
    this.getClass().getName(),
    "Entering applies()";

    boolean applies = false;
    File doc = busDoc.getDocument();
    BusinessObjectFactory bof = null;
    ItsoPackage pkg = null;
    try {
        bof = BusinessObjectFactory.getInstance();
        pkg =
            (ItsoPackage)bof.getBusinessObject(ExitConstants.ITSO_PACKAGE_CODE,
            ExitConstants.ITSO_PACKAGE_FORMAT);
        pkg.unmarshall(doc);
        if(pkg.getName().equals(ExitConstants.ITSO_PACKAGE_CODE) &&
            pkg.getVersion().equals(ExitConstants.ITSO_PACKAGE_VERSION)) {
            applies = true;
            busDoc.setTempObject(ExitConstants.TMP_BUSINESS_OBJ,pkg);
        }
    } catch (Exception e) {
        applies = false;
        bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        e.getClass().getName() + " : " + e.getMessage() + " : " +
        e.toString());
    }
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
    this.getClass().getName(),
    "Leaving applies()";
    return applies;
}
```

*Figure 19-40  The applies() method*

- **init()**: This method remains blank since our design did not require any specific initialization code.
**process()**: This method (Figure 19-41) decrypts the file and strips off the ITSO packaging headers. It relies on the fact that we uploaded certificates in the right profiles (see Chapter 8, “Securing the B2B exchange” on page 189).

```java
ItsoPackage pkg = null;
pkg = (ItsoPackage)busDoc.getTempObject(ExitConstants.TMP_BUSINESS_OBJ);
if(pkg == null){
    params = new String[1];
    params[0] = "BusinessObject was null";
    // Raise Event
} else {
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPACKAGINGCD, pkg.getName());
    busDoc.setAttribute(BCGDocumentConstants.BCG_FRPACKAGINGVER, pkg.getVersion());
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_INITBUSINESSID, pkg.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_FRBUSINESSID, pkg.getFrom());
    busDoc.setAttribute(BCGDocumentConstants.BCG_PKG_TOBUSINESSID, pkg.getTo());

    SecurityServiceInterface ssi = BusinessProcessUtil.getSecurityService();

    byte[] clearText;
    try {
        clearText =
            ssi.decryptBytes(
                busDoc,
                pkg.getPayload(),
                SecurityServiceInterface.BCG_ENCRYPT_ALG_3DES);
        File decryptedFile = busDoc.createFile();
        FileOutputStream fos = new FileOutputStream(decryptedFile);
        fos.write(clearText);
        fos.close();
        busDoc.setDocument(decryptedFile);

        } catch (Exception e){
            // Raise Event
        }
}
return busDoc;
```

*Figure 19-41 The process() method*
**ItsolInboundFWProcessor**

The ITSOLnboundFWProcessor is responsible for processing the custom ITSO package and setting the corresponding BCG headers. Its main methods are:

- **applies()**: This method (Figure 19-42) returns *true* if the document received was packaged in the ITSO package.

```java
public boolean applies(BusinessDocumentInterface busDoc) {
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Entering applies()";
    boolean applies = false;
    // Use get xml root to marshall the doc
    File doc = busDoc.getDocument();
    BusinessObjectFactory bof = null;
    BusinessObject bo = null;
    try {
        bof = BusinessObjectFactory.getInstance();
        BusinessObjectUtil bou = new BusinessObjectUtil();
        String process = (String)bou.getXmlRootElement(doc);
        bo = bof.getBusinessObject(process,
            ExitConstants.ITSO_XML_PROTOCOL_CODE
        );
        bo.unmarshall(doc);
        if(bo.getProtocolName().equals(ExitConstants.ITSO_XML_PROTOCOL_CODE)
            &&
            bo.getProtocolVersion().equals(ExitConstants.ITSO_XML_PROTOCOL_VERSION)) {
            applies = true;
        }
    } catch (Exception e) {
        applies = false;
        bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
            this.getClass().getName(),
            e.getClass().getName() + ": " + e.getMessage() + " : " +
            e.toString());
    }
    busDoc.setTempObject(ExitConstants.TMP_BUSINESS_OBJ,bo);
    bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
        this.getClass().getName(),
        "Leaving applies()";
    return applies;
}
```

*Figure 19-42  The applies() method*
- **init()**: This method remains blank since our design did not require any specific initialization code.

- **process()**: This method (Figure 19-43) sets the necessary BCG headers that WebSphere BI Connect requires by deriving this information from the ITSO headers.

```java
// job here is to set the following attributes
busDoc.setAttribute(BCGDocumentConstants.BCG_FRBUSINESSID,
   busDoc.getAttribute(BCGDocumentConstants.BCG_PKG_FRBUSINESSID));
busDoc.setAttribute(BCGDocumentConstants.BCG_TOBUSINESSID,
   busDoc.getAttribute(BCGDocumentConstants.BCG_PKG_TOBUSINESSID));
busDoc.setAttribute(BCGDocumentConstants.BCG_INITBUSINESSID,
   busDoc.getAttribute(BCGDocumentConstants.BCG_PKG_INITBUSINESSID));

// the rest of the headers will be obtained by processing the document
BusinessObject bo = null;
bo = (BusinessObject)busDoc.getTempObject(ExitConstants.TMP_BUSINESS_OBJ);

busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROTOCOLNAME,
   bo.getProtocolName());
busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROTOCOLVER,
   bo.getProtocolVersion());

busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROCESSCD,
   ExitConstants.ALL_VERSION);
busDoc.setAttribute(BCGDocumentConstants.BCG_FRPROCESSVER,
   ExitConstants.ALL_VERSION);

bcgUtility.trace(BCGUtil.BCG_TRACE_SEVERITY_DEBUG,
   this.getClass().getName(),
   "Leaving process()");
return busDoc;
```

*Figure 19-43   The process() method*
19.7 Deploying and testing the ITSO workflow

The following section provides detailed instructions about how these custom workflow components are introduced into the WebSphere BI Connect system. The instructions walk you through the necessary steps and relate the current configuration to the document flow definitions and interactions that were previously defined.

To continue our description of the Company A to Company E flow, we start with the configuration of the Company A WebSphere BI Connect instance and move on to the required configuration for Company E.

After we complete the required steps, we run an end-to-end test. This test involves all of the components for our use case and demonstrates the complete flow of our ITSO custom extensions.

19.7.1 Creating the user exits JAR file

The ITSO user exit code has to be distributed to both Company A and Company E. To distribute the JAR file, export the project from WebSphere Studio Application Developer.

1. Right-click the project in Studio and select Export as shown in Figure 19-44.
Figure 19-44   WebSphere Studio Application Developer Java perspective
2. In the Export window (Figure 19-45), select **JAR file** and click **Next**.
3. In the JAR Export window (Figure 19-46), select the appropriate packages and files. Provide a file name and click **Finish**.

![JAR Export window](image)

**Figure 19-46   JAR Export window**
19.7.2 Company A workflow deployment

After the source is exported, follow these steps:

1. Place the JAR file in the `<wbic_install>\was\wbic\userexits` directory, where the asterisk (*) represents the router and the receiver directory.

2. Copy the JAXME JAR files into the `<wbic_install>\router\was\java\lib` directory. These are needed by the business object framework.

3. Restart both the receiver and router for them to pick up the new JAR files.

User exit deployment descriptors

Accompanying the placement of the JAR files, the developer must create a deployment descriptor for each of the user exit components. For Company A, the development team created the following deployment descriptors:

- **CompanyaCsvInboundProcessor.xml**: This descriptor (Figure 19-47) describes the inbound fixed workflow. It provides the name of the implementation class, a description, and the handler type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"
bcgimport.xsd ">
<tns:HandlerClassName>com.ibm.itso.wbic.exits.CompanyaInboundProcessor</tns:HandlerClassName>
<tns:Description>Company A CSV Protocol Inbound Fixed Workflow</tns:Description>
<tns:HandlerTypes>
  <tns:HandlerTypeValue>FIXEDWORKFLOW.PROTOCOL.PARSE</tns:HandlerTypeValue>
</tns:HandlerTypes>
</tns:HandlerDefinition>
```

*Figure 19-47  Descriptor CompanyaCsvInboundProcessor.xml*
Chapter 19. Implementing workflow user exits

- **ItsoTransformationFactory.xml**: This descriptor (Figure 19-48) describes the variable workflow. It provides the name of the implementation class, a description, and the handler type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
cbgimport.xsd">
  <tns:HandlerClassName>com.ibm.itso.wbic.exits.TransformationFactory</tns:HandlerClassName>
  <tns:Description>Company A transformation factory to convert company A csv to ITSO xml</tns:Description>
  <tns:HandlerTypes>
    <tns:HandlerTypeValue>ACTION.TRANSFORMATION</tns:HandlerTypeValue>
  </tns:HandlerTypes>
</tns:HandlerDefinition>
```

*Figure 19-48  Descriptor ItsoTransformationFactory.xml*

- **ItsoPackager.xml**: This descriptor (Figure 19-49) describes the outbound fixed workflow. It provides the name of the implementation class, a description, and the handler type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
cbgimport.xsd">
  <tns:HandlerClassName>com.ibm.itso.wbic.exits.ItsoFWPackager</tns:HandlerClassName>
  <tns:Description>Custom unpackaging for ITSO WBI-C exits redbook</tns:Description>
  <tns:HandlerTypes>
    <tns:HandlerTypeValue>FIXEDWORKFLOW.PROTOCOL.PACKAGING</tns:HandlerTypeValue>
  </tns:HandlerTypes>
</tns:HandlerDefinition>
```

*Figure 19-49  Descriptor ItsoPackager.xml*

These deployment descriptors are in the descriptors folder in the WebSphere Studio Application Developer User Exits project, which you can download from the Web. The schemas that are used to create deployment descriptors are found in the folder schemas in the same project and are available on the product installation CD.
Inbound fixed workflow

Company A is required to create an inbound fixed workflow user exit to manage the documents received from its back-end application. To add the inbound handler to the Company A system, follow these steps:

1. Log into Company A as the hub administrator. Select Hub Admin → Hub Configuration → Handlers → Fixed Workflow. Click Import.

![List of fixed workflow handlers](image)
2. In the Import Handler window (Figure 19-51), complete these items:
   a. Provide the directory and name of the descriptor file
      CompanyaCsvInboundProcessor.xml.
   b. For Commit to database, select Yes.
   c. Click Upload.

Figure 19-51  Importing the new handler
When the import succeeds, it is listed at the end of the list of available handlers (Figure 19-52). Notice that provider is listed as User for this new handler. Also, be aware that user-provided handlers can be deleted, while product-provided handlers cannot be deleted.

![Fixed Workflow HandlerList](image)

*Figure 19-52  Fixed Workflow HandlerList*
Now add the handler to the inbound handler chain.

1. As shown in Figure 19-53, select **Hub Admin → Hub Configuration → Fixed Workflow → Inbound**.

2. Click the magnifying glass icon next to the entry for **ChannelParseFactory**.

---

**Figure 19-53**  Inbound fixed workflow step list
3. In the Step Details window (Figure 19-54), click the edit icon.

![Figure 19-54 ChannelParseFactory configuration](image)
4. In the next Step Details window (Figure 19-55), do these tasks:
   a. Select the newly imported handler in the list of available handlers and click Add.

---

*Figure 19-55  Editing the ChannelParseFactory configuration (Part 1 of 2)*
b. The new handler is added to the bottom of the list of configured handlers. Click **Move Up** until the handler is the first listed in the chain, as shown in Figure 19-56.

c. Click **Save**.

*Figure 19-56   Editing the ChannelParseFactory configuration (Part 2 of 2)*
Figure 19-57 shows the updated ChannelParseFactory configuration.
Variable workflow
The variable workflow required by Company A needs to be imported and configured.

1. Select **Hub Admin → Hub Configuration → Handlers → Action**.
2. When the list of action handlers is shown, click **Import**. See Figure 19-58.

![Figure 19-58 List of action handlers](image-url)
3. In the Import Handler window (Figure 19-59), complete these tasks:
   a. Provide the name and directory of the transformation handler
      ItsoTransformationFactory.xml.
   b. For Commit to database, select **Yes**.
   c. Click **Upload**.

**Figure 19-59  Import Action Handlers**
The new action handler is added at the bottom of the list as shown in Figure 19-60. As before, it is categorized as a user-provided handler and can be deleted.

<table>
<thead>
<tr>
<th>HandlerType</th>
<th>Classname</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION.DUPLICATECHECK</td>
<td>com.ibm.boq.duplicate.ContentDuplicateProcessFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.TRANSFORMATION</td>
<td>com.ibm.boq.outbound.OutboundCoreDslFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.TRANSFORMATION</td>
<td>com.ibm.boq.passsthrough.NoOp</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.TRANSFORMATION</td>
<td>com.ibm.boq.translation.protocol.JWTAndHmacProtFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.TRANSFORMATION</td>
<td>com.ibm.boq.translation.protocol.JWTAndRNSEncProtFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.VALIDATION</td>
<td>com.ibm.boq.translation.protocol.translators.wsl.xslt.XSLTTranslationFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.VALIDATION</td>
<td>com.ibm.boq.validation.OutboundValidationFactory</td>
<td>Product</td>
</tr>
<tr>
<td>ACTION.VALIDATION</td>
<td>com.ibm.boq.validation.ValidationFactory</td>
<td>Product</td>
</tr>
</tbody>
</table>

Figure 19-60  Action Handlers List
4. Associate the new handler to an action. As shown in Figure 19-61, select **Hub Admin → Hub Configuration → Actions**.

5. When the list of actions is shown, click **Create**.

![WebSphere Business Integration Connect Community Console](image_url)

**Figure 19-61 List of actions**

6. Create the action so it can be associated with the interaction created in 19.3.2, "Creating an interaction" on page 616. In the Actions window (Figure 19-62), complete these tasks:
a. Provide a name for the new action, such as ITSOTransformationFactory.
b. Provide a description.
c. Select the **Enabled** option.
d. Select the custom handler that was imported. The handler should be at the bottom of the list of available handlers.
e. Click **Add**.
f. Click **Save**.

---

**Figure 19-62**  Creating a new action
The new action should be listed at the bottom of the list of available actions as shown in Figure 19-63. As before, it should be labeled as a user-provided action and can be deleted.

Figure 19-63  Updated list of actions
To associate the action to the interaction, let’s go back to the interaction defined in 19.3.2, “Creating an interaction” on page 616, and replace the Pass Through action with the new action ItsoTransformationFactory.

2. Click Manage Interactions.
3. In the Manage Interactions window (Figure 19-64), search for the interaction created before by providing a package and protocol. Click the magnifying glass icon to edit it.

![Manage Interactions](image-url)
4. A window opens that allows you to make changes to the interaction. See Figure 19-65. Select the action **ITSOTransformationFactory** from the list and click **Save**.

![Edit Interactions](image)

*Figure 19-65  Edit Interactions*
**Participant connection**

Now that you have updated the interaction to make use of our custom action, you can create a participant connection to represent this channel.

1. Select **Account Admin → Participant Connections**.
2. As shown in Figure 19-66, select the source and target companies and click **Search**.

![WebSphere Business Integration Connect Community Console](image)

_Figure 19-66  List of participant connections between Company A and Company E_
3. The Manage Connections window (Figure 19-66) opens, showing the existing and activated connections. These connections were used previously to exchange EDI documents. A new connection appears as well, which is currently not activated. Click **Activate**.

![Manage Connections screenshot](image)

**Figure 19-67  Participant connection is activated**
Outbound Fixed Workflow

The outbound fixed workflow handler is imported in the same way as the inbound fixed workflow. To add the package handler to the system, follow these steps:

1. Select Hub Admin → Hub Configuration → Handlers → Fixed Workflow.
2. When you see the list of handlers, click Import to import the deployment descriptor.
3. In the Import Handler window (Figure 19-68), complete these tasks:
   a. Provide the name and directory of the descriptor file ItsoPackager.xml.
   b. For Commit to database, select Yes.
   c. Click Upload.

![Figure 19-68 Import Handler](image-url)
The new handler is added at the bottom of the list. Add the new handler to the outbound fixed workflow chain to package the ITSO document.

1. Select **Hub Admin → Hub Configuration → Fixed Workflow → Outbound**.
2. Click the magnifying glass icon next to the outbound fixed workflow called **ProtocolPackagingFactory**. See Figure 19-69.

![Figure 19-69 Outbound Fixed Workflow Step List](image)
3. In the Step Details window (Figure 19-70), complete these tasks:
   a. Click the edit icon.
   b. Add the new handler from the list of available handlers to the list of configured handlers.
   c. Click **Move Up** to move the custom package handler to the top of the list.
   d. Click **Save**.

![Figure 19-70   Editing the ProtocolPackagingFactory configuration](image-url)
Figure 19-71 shows the updated ProtocolPackagingFactory configuration. This completes the deployment and configuration of the user exits for Company A.

19.7.3 Company E workflow deployment

After the source is exported, place the JAR file in the `<wbic_install>\router\was\wbic\userexits` directory of the WebSphere BI Connect server of Company E. You must restart the router to pick up the new JAR file.

**User exit deployment descriptors**

Accompanying the placement of the JAR files, the developer must create a deployment descriptor for each of the user exit components. For Company E, the development team created the following deployment descriptors:
- **ItsoUnpackager.xml**: This descriptor (Figure 19-72) describes the unpackaging portion of the inbound fixed workflow. It provides the name of the implementation class, a description, and the handler type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
xmllns:xsi="http://www.w3.org/2001/XMLSchema-instance"
bcgimport.xsd " >
  <tns:HandlerClassName>com.ibm.itso.wbic.exits.ItsoFWUnpackager</tns:HandlerClassName>
  <tns:Description>Custom unpackaging for ITSO WBI-C exits redbook</tns:Description>
  <tns:HandlerTypes>
    <tns:HandlerTypeValue>FIXEDWORKFLOW.PROTOCOL.UNPACKAGING</tns:HandlerTypeValue>
  </tns:HandlerTypes>
</tns:HandlerDefinition>

Figure 19-72  Descriptor ItsoUnpackager.xml

- **ItsoInboundProcessor.xml**: This descriptor (Figure 19-73) describes the protocol parsing portion of the inbound fixed workflow. It provides the name of the implementation class, a description, and the handler type.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:HandlerDefinition
xmllns:xsi="http://www.w3.org/2001/XMLSchema-instance"
bcgimport.xsd " >
  <tns:HandlerClassName>com.ibm.itso.wbic.exits.ItsoInboundFWProcessor</tns:HandlerClassName>
  <tns:Description>Inbound fixed workflow protocol processing handler for itso wbic redbook</tns:Description>
  <tns:HandlerTypes>
    <tns:HandlerTypeValue>FIXEDWORKFLOW.PROTOCOL.PARSE</tns:HandlerTypeValue>
  </tns:HandlerTypes>
</tns:HandlerDefinition>

Figure 19-73  Descriptor ItsoInboundProcessor.xml

You can find these deployment descriptors in the descriptors folder in the WebSphere Studio Application Developer User Exits project, which you can
download from the Web. Look for the schemas used to create the deployment descriptors in the schemas folder in the same project. They are also available on the installation CD of the product.

**Inbound Fixed Workflow**
Company E needs to create an inbound fixed workflow user exit to manage the documents that are sent by Company A. To add the inbound handler to the system of Company E, follow these steps:

1. Log into Company E as the hub administrator. Select **Hub Admin ➔ Hub Configuration ➔ Handlers ➔ Fixed Workflow**.

2. When the list of handlers is shown, click **Import**.

3. In the Import Handler window (Figure 19-74), provide the directory and the name of the descriptor file, **ItsoUnpackager.xml**, and click **Upload**.

![Figure 19-74 Importing the fixed workflow handler ItsoUnpackager.xml](image-url)
4. Import the second deployment descriptor, ItsoInboundProcessor.xml, in the same way as shown in Figure 19-75.

![Import Handler](image)

*Figure 19-75  Importing the fixed workflow handler ItsoInboundProcessor.xml*
The two new handlers are added at the bottom of the list of handlers, as shown in Figure 19-76.

<table>
<thead>
<tr>
<th>HandlerType</th>
<th>Classname</th>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.xml.XMLChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.xml.XMLPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.esi.EAIPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.esi.EAIUnPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.edi.EDIRouterBizProcessHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.edi.EDIUnPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.BinaryChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.RNOChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.RNOChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.RNOChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.RNOChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.XMLRouterBizProcessHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.xml.MimeMultipartUnpackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.xml.NullUnPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.xml.RNFPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.soap.SOAPChannelParseHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.soap.SOAPPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.server.NullPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.beg.edi.EDIPackagingHandler</td>
<td>Product</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.ie.so.wbic.exits.ItsoFWUnpackager</td>
<td>User</td>
</tr>
<tr>
<td>FIXEDWORKFLOW, PROTOCOLPARSE</td>
<td>com.ibm.ie.so.wbic.exits.ItsoInboundFWProcessor</td>
<td>User</td>
</tr>
</tbody>
</table>

Figure 19-76  List of fixed workflow handlers
Now associate the new handlers to the inbound workflow steps.

1. Select **Hub Admin → Hub Configuration → Fixed Workflow → Inbound**.

2. Two workflow steps should be listed. Click the magnifying glass icon for **TransportUnpackagingFactory**. See Figure 19-77.

---

**Figure 19-77   Inbound Fixed Workflow Step List**

<table>
<thead>
<tr>
<th>Step name</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.ibm.bcg.server.transport.TransportUnPackagingFactory</td>
</tr>
<tr>
<td>com.ibm.bcg.server.ChannelParseFactory</td>
</tr>
</tbody>
</table>
3. Follow the steps outlined in the Company A configuration to set the ItsoUnpackager in the first position of the TransportUnpackageFactory.
   a. When you see the details of this step, click the edit icon.
   b. Move the handler **ItsoFWUnpackager** from the list of available handlers to the list of configured handlers.
   c. Click **Move Up** to position this handler at the top of the list.
   d. Click **Save**.

   Figure 19-78 shows the updated inbound fixed workflow step TransportUnpackagingFactory.
4. Follow similar steps to add the handler `ItsoInboundProcessor` in the first position of the inbound fixed workflow step `ChannelParseFactory`, as shown in Figure 19-79.

![Updated ChannelParseFactory configuration](image)
Participant connection
Now that we imported the custom workflow components, we can create the participant connection to represent this Company A to Company E channel.

1. Select **Account Admin → Participant Connections**.
2. Select the source and target companies and click **Search**.
3. You see the existing and activated connections as shown in Figure 19-80. These connections were used previously to exchange EDI documents. A new connection appears as well, which is currently not activated. Click **Activate**.

![Figure 19-80   List of participant connections](image)
4. After the connection is activated, click **Gateways** as shown in Figure 19-81.

![Figure 19-81  Participant connection](image-url)
5. In the Edit Participant Connection window (Figure 19-82), for Target Gateways, select the custom sender ApplicationFileNamingSender and click Save.

![Edit Participant Connection window](image)

**Figure 19-82  Setting the target gateway for a new connection**

**Note:** In 18.5.3, “Sender configuration” on page 573, we altered the target gateway for the interaction between AS2/EDI-X12 and None/EDI-X12. This allowed us to test the custom sender while we continued to use the EDI document flow. At this time, we can reset the target gateway for the EDI document flow to the JMSGateway or to the FileSystemGateway.

The systems are now configured to send and receive custom ITSO documents.
19.8 Testing the flow

To test the flow, we use similar steps as in Chapter 18, “Implementing receiver and sender user exits” on page 535. This time the document flow moves through each of our custom components. To execute this test, we use a text file that contains an invoice with comma-separated elements. A sample file, TestInvoice.csv, is provided as part of the downloadable materials for this redbook. It is also shown in Figure 19-83.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestInvoice.csv</td>
<td>Sample comma-separated invoice</td>
</tr>
</tbody>
</table>

Before you run the test, make sure that the WebSphere BI Connect components (receiver and router on both Company A and E) are started. We recommend that you also delete the current log files of both servers. That makes it easier to locate errors during the execution.

Now follow these steps:

1. On the server of Company A, copy the file with extension.csv in the directory /opt/IBM/WBIConnect/data/companya/FileMatching, which is the directory polled by the custom receiver of Company A.

2. On the next poll cycle, the TestInvoice.csv file is picked up and the document flow is executed. After the TestInvoice.csv file is picked up, log into the console for Company A and navigate to the Document Viewer.

3. Execute a query for the latest flow. You should see a document exchange that represents the exchange of ITSO documents, as shown in Figure 19-84. Click the document icon that represents the source document.
You should now see the document in the raw document viewer.

### Figure 19-84  Company A Document Viewer

You should now see the document in the raw document viewer.

<table>
<thead>
<tr>
<th>Participants</th>
<th>Time Stamps</th>
<th>Document Flow</th>
<th>Gateway Type</th>
<th>Synchronous</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source: Company A</td>
<td>In: 11/19/04 1:14:22 AM</td>
<td>COMPA_CSV (1.0) ALL: ALL (ALL)</td>
<td>Production</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target: Company E</td>
<td>Out: 11/19/04 1:14:24 AM</td>
<td>ITSO_XML (1.0) ALL: ALL (ALL)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 19-85  Source document**

4. In Figure 19-84, click the document icon that represents the target document.
You should now see a document with an HTTP transport header and an encrypted document (Figure 19-86). Notice that the ITSO package is not encrypted.

5. After you verify that Company A has sent the file, log into the Company E console and navigate to the Document Viewer.

6. Execute a query for the latest flow.
7. You should see a document exchange representing the exchange of ITSO documents, as shown in Figure 19-87. Click the document icon that represents the source document.

Figure 19-87   Company E document viewer
You should now see the document in the raw document viewer as shown in Figure 19-88. This is the decrypted version of the document.

<table>
<thead>
<tr>
<th>Participants</th>
<th>Business IDs</th>
<th>Document Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source: Company A</td>
<td>companyA</td>
<td>ITSO_XML (1.0) ALL: ALL (ALL)</td>
</tr>
<tr>
<td>Target: Company E</td>
<td>companyE</td>
<td>ITSO_XML (1.0) ALL: ALL (ALL)</td>
</tr>
</tbody>
</table>

Figure 19-88  Decrypted document that is received by Company E
8. In Figure 19-87 on page 687, click the document icon that represents the target document.

You should now see the XML document that represents the invoice as shown in Figure 19-89. HTTP packaging is removed as is the ITSO packaging.

![XML Document](image)

**Figure 19-89 Unpackaged document received by Company E**
9. A final test is to verify that the XML document is written by the custom sender in the correct directory.

a. Open Windows Explorer.

b. Navigate to the directory that is configured on the custom sender, C:\WBIC\Data\FileNamingSender.

You should see a file with the .xml extension. This file should contain the invoice document that was also shown in the document viewer.

Figure 19-90   Contents of the directory used by the custom sender
Implementing RosettaNet in WebSphere BI Connect
Introduction to RosettaNet

This chapter discusses the RosettaNet e-business process standard and provides a brief overview describing its evolution over time. It also discusses IBM's involvement with the RosettaNet organization and our interoperability status.
20.1 A brief history

RosettaNet is a non-profit consortium dedicated to the collaborative development and rapid deployment of open, Internet-based business standards that align processes within the global trading network. More than 500 multinational and regional companies in the consumer and computer electronics (CCE), electronic components (EC), logistics (LG), semiconductor manufacturing (SM), and telecommunications (TC) industries participate in RosettaNet's strategic standards and services development. Fortune 1000 companies worldwide have implemented RosettaNet standards, conducting B2B operations that value billions in annual revenues.

RosettaNet is one of the industry's most ambitious standards implementation efforts. To date, the consortium has established several regional affiliate organizations, in the Americas, Europe, Australia, China, Japan, Korea, Malaysia, Philippines, Singapore, and Taiwan. Each gives a voice to various business economies seeking to adopt and influence RosettaNet's global standards. You can find information about RosettaNet's worldwide activities, including a complete list of member companies and participating organizations, on the Web at:

http://www.RosettaNet.org

RosettaNet is a subsidiary of the Uniform Code Council, Inc.® (UCC®).

20.1.1 IBM's involvement with the RosettaNet organization

Several industry councils make up the RosettaNet organization:

- The Consumer and Computer Electronics Council consists of leading representatives of the computer and consumer electronics industry. Participants in the CCE Council include Original Equipment Manufacturers (OEM), distributors, retailers, and service providers.

- The Semiconductor Manufacturing Council consists of leading representatives of the semiconductor manufacturing industry. Participants in the SM Council include such companies as fabless device manufactures, foundries, materials suppliers, and assembly and test providers.

- The Electronic Components Council consists of leading representatives of the electronics components industry. Participants of the EC Council include such companies as semiconductor suppliers, distributors, connector suppliers, and cable suppliers.

- The Logistics Council consists of leading representatives of the logistics sector. Participants in the LG Council includes semiconductor suppliers, passive suppliers, connector suppliers, distributors, and customers.
The Solution Provider Council consists of leading representatives of the solution provider industry. Participants in the SP Council include such companies as solution integrators, consulting services, and software development services.

The Telecommunications Council consists of leading representatives of the telecommunications industry. Participants in the TC Council include such companies as global network providers and equipment suppliers.

In addition to the councils, the RosettaNet organization has a partnership program, which currently has a membership count of over 500 companies from around the world. Membership levels from the highest tier to the lowest tier are Premier Partner, Partner, and Associate Partner. Only Premier Partners can participate in RosettaNet's Global Industry Councils.

IBM is a Premier Partner of RosettaNet and actively participates in the CCE, SM, EC, and SP Councils. As a Premier Partner, IBM continues to work with other council members to promote RosettaNet e-business process standards adoption within each industry.

### 20.1.2 Interoperability

Interoperability between partners is a significant business problem for many companies who want to trade with their partners who are using the RosettaNet e-business process standard. Software vendors create B2B solutions that are capable of supporting RosettaNet. The interpretation of the specification often differs between vendors. This causes issues when trading RosettaNet documents with partners who use disparate B2B solutions.

To increase RosettaNet adoption and to decrease the chance of interoperability issues, the RosettaNet organization has developed a set of development tools and management process that measure compliance for software products. This program is called RosettaNet Ready. It is a subscription service that provides members with testing tools, compliance testing methodology and badges to identify products that have successfully completed compliance testing. IBM is among the list of vendors who have completed the RosettaNet Ready program.

RosettaNet Ready is the legacy program for RosettaNet software compliance testing. Software products tested as compliant under this program are designated by the RosettaNet Ready Badge. In October 2004, compliance testing and certification became available through the UCC Solution Partner Program. This program offers software compliance testing through the eBusinessReady® program. RosettaNet Ready and eBusinessReady compliance certifications are both endorsed and recognized by RosettaNet.
Compliance by itself is not enough to ensure interoperability between software solutions. Compliance is different than interoperability. It ensures that the requirements of the standard are adhered to, decreasing the chance of interoperability issues. Interoperability testing is designed to ensure that disparate software solutions can work with each other. When a product is certified as interoperable, the customer can be reasonably sure that they can trade RosettaNet documents with any vendor who has also completed the interoperability testing process.

RosettaNet interoperability testing is provided by eBusinessReady, an industry-neutral testing program for e-business software vendors that was launched in November 2001 to enable vertical and horizontal compliance and interoperability across the supply chain and distribution channels. The program, which is a joint partnership of the UCC and Drummond Group Inc. (DGI), brings together software vendors, exchanges, vertical industries, and the standards community in a neutral, standards-based testing environment that simulates real-world operating conditions.

For more information about eBusinessReady, refer to the program’s Web site at: http://www.eBusinessReady.org

20.1.3 Electronic business exchange

RosettaNet standards are analogous to human-to-human business exchange, which typically is successful and efficient because business partners agree upon the process from the most basic level. For example, we produce and hear sound, use a common alphabet to create words, apply grammatical rules to words to make dialog, use dialog to form business processes, and conduct business through an instrument such as a telephone.

In an automated environment, RosettaNet uses servers to exchange information over the Internet. HTML and XML function as the alphabet, and electronic commerce applications serve as the instrument by which B2B processes are transmitted. The lack of agreement on the words, grammar and dialog that constitute B2B processes illustrates the need for standards. RosettaNet dictionaries provide the words, the RosettaNet Implementation Framework (RNIF) acts as the grammar, and RosettaNet Partner Interface Processes (PIPs) form the dialog.

To conduct B2B within a single supply chain, there needs to be four XML-related components present as defined by the RosettaNet standard: Business Processes, Technical Dictionary Structure, Business Dictionary Structure and a Messaging Service as illustrated in Figure 20-1.
The PIPs define the business process, which specifies the structure and format of the business content of a message. The RosettaNet Dictionary Framework provides the common set of components and conventions for the content of the business message. RNIF is the foundation of the stack. It specifies how the information is physically packaged, transferred, and routed securely between trading partners.

### 20.2 Partner Interface Process

PIPs are either dual action or single action. **Dual action PIPs** are used in cases where a two-way conversation is needed to process the information that is being sent. A good example of this process is a buyer and seller scenario. The buyer wants to order 10 widgets, so he sends a purchase order request (PIP3A4 - Request Purchase Order Action) to the seller of the widgets. The seller sends back a purchase order confirmation (PIP3A4 - Purchase Order Confirmation action), advising the buyer that they are processing the buyer’s order.

**A single action PIP** involves only a one-way conversation. It is used in cases where a response is not needed. A good example of this process is where the seller is sending an inventory report to the buyer. The seller sends an Inventory Report (PIP4C1) to the buyer to inform them of the status of their inventory. No return action is necessary since the seller already knows the status.

The PIPs are divided into seven clusters of core business processes with each being broken down into segments. Segments contain individual PIPs. Figure 20-2 illustrates how PIPs are laid out.
For a complete list of PIPs, see the RosettaNet Web site:
http://www.rosettanet.org/RosettaNet/pips

20.2.1 Cluster 0: RosettaNet Support

Cluster 0 provides administrative functionality. It consists of two segments:

- Segment 0A is used for administrative purposes.
- Segment 0C is intended for asynchronous and synchronous testing.

20.2.2 Cluster 1: Partner Product and Service Review

Cluster 1 allows for information collection, maintenance and distribution for the development of trading partner profiles. It consists of two segments:

- Segment 1A is used for partner review and provides the ability to share information.
- Segment 1B is used for product and service review, and allows suppliers to manage product information using a subscription process.
20.2.3 Cluster 2: Product Information

Cluster 2 allows distribution and updates of product and design information. It consists of four segments:

- Segment 2A is used to distribute and query product information. This includes information about sales, marketing, design engineering, stock keeping units (SKU), life cycle and technical information.
- Segment 2B is used for the update of product resources.
- Segment 2C is used for release and update of product engineering design information.
- Segment 2D is used to allow collaborative design and engineering of new products and revisions to the new products. Currently no PIPs are available for this segment.

20.2.4 Cluster 3: Order Management

Cluster 3 supports the entire order management process. It consists of four segments:

- Segment 3A is used for order quotes and status and purchase order information.
- Segment 3B is used to communicate shipping and delivery information. It also provides the ability to make changes and handle claims and exceptions.
- Segment 3C is used for billing information, payment and reconciliation of debits, credits and invoices between partners.
- Segment 3D is used for product configuration that feeds the order management process.

20.2.5 Cluster 4: Inventory Management

Cluster 4 supports the entire inventory management process. It consists of six segments:

- Segment 4A is used for sharing forecasting information between partners.
- Segment 4B is used for inventory allocation, allowing sellers to inform buyers about product allocation.
- Segment 4C is used for inventory reporting, providing sellers with daily inventory reports.
- Segment 4D is used for inventory replenishment for both the buyer and seller.
- Segment 4E is used for sales reporting from buyers to product providers.
Segment 4F is used to announce and acknowledge price changes and to process payments or credits as a result of the changes. Currently no PIPs are available for this segment.

20.2.6 Cluster 5: Marketing Information Management

Cluster 5 provides communications of marketing information. It consists of four segments:

- Segment 5A is used for lead opportunity management for sharing sales lead information. Currently no PIPs are available for this segment.

- Segment 5B is used for marketing campaign management, enabling the communication of marketing campaign details. Currently no PIPs are available for this segment.

- Segment 5C is used for design win management, enabling design registration with suppliers.

- Segment 5D is used to query, create and notify based on ship from stock and debit information.

20.2.7 Cluster 6: Service and Support

Cluster 6 provides technical support, warranty and asset management capabilities. It consists of three segments:

- Segment 6A enables registration and product warranty support. Currently no PIPs are available for this segment.

- Segment 6B provides and administers asset management. Currently no PIPs are available for this segment.

- Segment 6C is used for technical support and service management.

20.2.8 Cluster 7: Manufacturing

Cluster 7 supports manufacturing environments. It consists of three segments:

- Segment 7A is used for the transfer of design information to manufacturing. Currently no PIPs are available for this segment.

- Segment 7B is used for the release, management and the exchange of factory production information.

- Segment 7C is used to distribute manufacturing information to support product improvements.
20.3 RosettaNet Dictionary Framework

RosettaNet Dictionary Framework provides a common vocabulary for conducting B2B. It eliminates confusion related to a companies’ usage of their own terminology. RosettaNet has developed two dictionaries that serve as a guideline for companies wanting to use RosettaNet for trading with their partners.

20.3.1 RosettaNet Business Dictionary

The RosettaNet Business Dictionary (RDBD) defines the business properties, business data entities and fundamental business data entities in PIP message guidelines. It designates the properties used for basic business activities.

20.3.2 RosettaNet Technical Dictionary

The RosettaNet Technical Dictionary (RNTD) provides the common language for defining products and services. It eliminates the need for trading partners to use separate dictionaries when implementing multiple PIPs.

20.4 RosettaNet Implementation Framework

The RosettaNet Implementation Framework is a messaging specification that defines the packaging, routing and transportation of RosettaNet messages. It includes message security and authentication. The framework provides implementation guidelines for the development of interoperable software in a core specification document.

There are two versions of the RNIF. RNIF version 1.1 is the legacy specification. RNIF version 2.0 is the current specification. For the purpose of this redbook, we briefly cover RNIF 1.1 and go into detail about RNIF 2.0.
20.4.1 RNIF Version 1.1

RNIF 1.1 was the first production release of the RosettaNet messaging specification. It has been well adopted in the high technology marketplace and still remains in use in many production environments. The specification defines a custom envelope with only signature security. Inside the envelope, you have a preamble header, delivery header and service content. The transport protocol used is HTTP or HTTPS. Figure 20-3 illustrates a typical RosettaNet Business Message for RNIF 1.1.

Figure 20-3  RNIF v1.1 packaging

The preamble and delivery header guidelines are defined in document type definition (DTD) files. Each header has its own guideline: PreamblePartMessageGuideline.dtd for the preamble and ServiceHeaderPartMessageGuideline.dtd for the service header.

The service content is defined by the PIP being used which also has DTDs that define how to implement the PIP.

20.4.2 RNIF version 2.0

RNIF 2.0 is a major improvement over RNIF 1.1. It supports both encryption and signing of the service header and service content using S/MIME enveloping. It allows the usage of multiple transport protocols, non-RosettaNet service content, and the synchronous exchange of request and response messages.

Figure 20-4 illustrates a typical RosettaNet Business Message for RNIF 2.0.
The preamble, delivery header and service header guidelines are defined in DTD much like RNIF 1.1. The service content is defined by the PIP being used, which also has DTDs that define how to implement the PIP. In addition to the PIP message, non-RosettaNet business documents can optionally be attached to the business message in the services content section.

RNIF 2.0 is not backward compatible with RNIF 1.1. Both partners must use the same version of the specification when trading RosettaNet messages with each other.

### 20.4.3 RosettaNet business message (RNIF v2.0)

This section briefly describes each section of a RNIF 2.0 business message. For a detailed description of each section, refer to the RNIF Core Specification v02.00.01, which you can download from the RosettaNet Web site at:

http://www.rosettanet.org
Preamble
The preamble identifies the standard with which the message structure is compliant. A DTD is available from RosettaNet that defines how the preamble is to be structured.

Delivery header
The delivery header identifies the sender and receiver, as well as information about the message instance. A DTD is available from RosettaNet that defines how the preamble is to be structured.

Service header
The service header identifies the PIP, the PIP instance, the activity and the action to which the message belongs. A DTD is available from RosettaNet that defines how the preamble is to be structured.

Service content
The service content contains XML business content and is always either a signal or action message. The DTD for the PIP action messages define how the service content is to be structured.

Attachments
Payloads that contain an action message can also contain attachments. Attachments do not need to be XML documents and usually support the business document. Attachments to the service content are sent as separate MIME body parts within the RosettaNet message. This provides packaging and shipping of the business content and attachments together. Attachments can also be referred to within the service content. They require the “Content-ID” MIME header element in the business message for each attachment.

RosettaNet business message security
RosettaNet uses the enveloped and signed data types in the specification Internet Engineering Task Force (IETF) RFC 2311 “S/MIME Version 2 Message Specification” as a guideline for RNIF Security. RNIF 2.0 recommends the usage of X.509 digital certificates for enveloping and signing and uses the specification IETF RFC 2312 “S/MIME Version 2 Certificate Handling” as a guideline for handling certificates. For more information about S/MIME specifications, refer to the IETF Web site at:

http://www.ietf.org
Implementing a RosettaNet solution

This chapter describes the process for implementing RosettaNet in WebSphere BI Connect. It uses an existing Partner Interface Process (PIP) as an example for the purpose of this scenario.

The chapter assumes that Company E and Company A have a working B2B environment and can trade encrypted and signed documents via AS2.
21.1 WebSphere BI Connect RosettaNet overview

You may have noticed that the application windows presented in this redbook, which show document flow definitions, do not have any information about the RosettaNet Implementation Framework (RNIF) packaging or the RosettaNet protocol. For example, look at Figure 11-6 on page 290. The reason is that there are two versions of the RNIF packaging (V1.0 and V1.1) and several versions of each PIP. WebSphere BI Connect can support only one combination of versions.

To support RosettaNet messaging, WebSphere BI Connect provides two sets of ZIP files called packages. The first set, the RNIF packages, consists of document flow definitions required to support the RNIF packaging. These packages are available in the B2BIntegrate directory of the WebSphere BI Connect installation CD.

- For RNIF V1.1
  - Package_RNIF_1.1.zip
  - Package_RNSC_1.0_RNIF_1.1.zip
- For RNIF V02.00
  - Package_RNIF_V02.00.zip
  - Package_RNSC_1.0_RNIF_V02.00.zip

The first package in each pair provides the document flow definitions required to support RosettaNet communications with participants. The second package provides the document flow definitions required to support RosettaNet communications with back-end systems.

The second set of packages consists of PIP document flow packages. Each PIP document flow package has a Packages directory. This directory contains an XML file and a GuidelineMaps directory containing XML Schema Definition Language (XSD) files. The XML file specifies the document flow definitions that define how WebSphere BI Connect handles the PIP and defines the exchanged messages and signals. The XSD files specify the format of the PIP messages and define acceptable values for XML elements in the messages. The ZIP files for 0A1 PIPs also have an XML file that the hub uses as a template to create 0A1 documents.

The PIPs for which WebSphere BI Connect provides PIP document flow packages are:

- PIP 0A1 Notification of Failure V1.0
- PIP 0A1 Notification of Failure V02.00.00
- PIP 2A12 Distribute Product Master V01.03.00
- PIP 3A1 Request Quote V02.00.00
- PIP 3A2 Request Price and Availability R02.01.00B
For each PIP, there are four PIP document flow packages:

- For RNIF 1.1 messaging with participants
- For RNIF 1.1 messaging with back-end systems
- For RNIF 2.0 messaging with participants
- For RNIF 2.0 messaging with back-end systems

Each PIP document flow package follows a specific naming convention so that you can identify whether the package is for messages between WebSphere BI Connect and participants or between WebSphere BI Connect and back-end systems. The naming convention also identifies the RNIF version, PIP, and PIP version that the package supports. For PIP document flow packages used for messaging between WebSphere BI Connect and participants, the format is:

BCG_Package_RNIF<RNIF version>_<PIP>_v<PIP version>.zip
For PIP document flow packages used for messaging between WebSphere BI Connect and back-end systems, the format is:

BCG_Package_RNSC<Backend Integration version>_RNIF<RNIF version>_<PIP><PIP version>.zip

For example, the BCG_Package_RNIF1.1_3A4V02.02.zip is for validating documents for version 02.02 of the 3A4 PIP sent between participants and WebSphere BI Connect using the RNIF 1.1 protocol. For PIP document flow packages for communicating with back-end systems, the name of the package must also identify the protocol used to send the RosettaNet contents to the back-end systems. See the Enterprise Integration Guide for information about the packaging used for these messages.

21.2 Trading scenario overview

In this scenario we work with the two WebSphere BI Connect instances that we implemented in the previous chapters. The first one is running WebSphere BI Connect Advanced on AIX. The second one is running WebSphere BI Connect Enterprise on Windows 2000. This chapter describes the steps to implement the exchange of RosettaNet documents between both partners. Each partner is given a role. Company E is the seller, and Company A is the buyer. We use PIP 3A4 in this scenario. Since a 3A4 PIP is dual action PIP, it can help to test communications in both directions.

Both partners use back-end integration packaging and use Java Message Service (JMS) as the integration protocol. We simulate the back-end systems by placing a PIP 3A4 Request RosettaNet Service Content (RNSC) document onto a JMS queue and passing it to the buyer’s WebSphere BI Connect for packaging, validation and transmission to the seller. The seller in turn places a PIP 3A4 Confirmation RosettaNet Service Content document onto a JMS queue, and passes it to the seller’s WebSphere BI Connect for packaging, validation and transmission to the buyer. This scenario completes both actions of the dual action PIP.

We use the IBM RFHUTIL utility to pass the RosettaNet Service Content and metadata to a JMS queue residing on the MQ queue manager that we created during installation. RFHUTIL is part of SupportPac IH03, which you can obtain from:


Keep in mind that, in most RosettaNet implementations, there are additional back-end systems that use the RosettaNet Business Messages. It is the
back-end system’s responsibility to send a confirmation or response action when using dual action PIPs. It is also the originator of the messages.

### 21.2.1 PIP 3A4 document flow (dual action PIP)

The document flow for this scenario is different from how it works in most RosettaNet implementations. A majority of companies have an Enterprise Application Integration (EAI) solution that connects WebSphere BI Connect to back-end applications such as SAP, Siebel, and Peoplesoft. To make this scenario less complicated, we created 3A4 request and confirmation files that contain the RosettaNet Service Content and the required metadata that can be loaded into the RFHUTIL utility.

Figure 21-1 shows a visual representation of the document flow.
First action: PO request (buyer to seller)
The first action of this document follows this sequence:

1. It begins by loading a PIP 3A4 request file into RFHUTIL and sending the file to the JMS target (Company_A_RN_OUT) on the buyer's system. The receiver component picks up the message and passes it to the document manager.

2. The PO request is picked up and processed by the document manager component. For outbound processing, we use an interaction that validates the PO request and transforms it to RNIF before passing it to the gateway.

3. The PO request is passed to the seller's outbound HTTP gateway (Company_E_RN_OUT) and sent over the Internet to the seller's system.

4. The PO request is received at the seller's HTTP target (Company_E_RN_IN) and is written into a directory polled by the document manager.

5. The PO request is picked up and processed by the document manager component. For inbound processing, we use an interaction that unpackages the PO request and transforms it to RosettaNet Service Content before passing it to the gateway.

6. The PO request is passed to the seller's JMS Gateway (Company_E_RN_IN) and persisted into a queue as RosettaNet Service Content.

   This message needs to be processed by an EAI system, the back-end systems of the seller, or both. However, in this chapter, we limit the RosettaNet solution at this level.

Second action: PO confirmation (seller to buyer)
The second action follows this sequence:

7. It begins by loading a PIP 3A4 confirmation file into RFHUTIL and sending the file to the JMS Target (Company_E_RN_OUT) on the seller's system. The receiver component picks up the file and writes it into a directory polled by the document manager.

8. The PO confirmation is picked up and processed by the document manager component. For outbound processing, we use an interaction that validates the PO confirmation and transforms it to RNIF before passing it to the gateway.

9. The PO confirmation is passed to the buyer's outbound HTTP gateway (Company_A_RN_OUT) and sent over the Internet to the buyer's system.

10. The PO confirmation is received at the buyer’s HTTP target (Company_A_RN_IN) and is written into a directory polled by the document manager.

11. The PO confirmation is picked up and processed by the document manager component. For inbound processing, we use an interaction that unpackages
the PO confirmation and transforms it to RosettaNet Service Content before passing it to the gateway.

12. The PO confirmation is passed to the buyer’s JMS Gateway (Company_A_RN_IN) and persisted into a queue as RosettaNet Service Content.

This message needs to be processed by an EAI system, the back-end systems of the buyer, or both. However, in this chapter, we limit the RosettaNet solution at this level.

### 21.3 Configuration tasks for hubadmin of Company E

This section discusses many of the tasks that are required for setting up RosettaNet in WebSphere BI Connect. The hubadmin must upload RosettaNet packages because the hub administrator is the only user that has the rights to upload packages into the system. The admin and the participant users can perform the remaining tasks much in the same way demonstrated in Chapter 7, “Creating a basic B2B exchange” on page 121. For the purpose of this scenario, we perform all of the RosettaNet configuration tasks on Company E’s system as the hubadmin user.

#### 21.3.1 Setting up MQ queues

Since we do not perform any integration with back-end systems, we can simply use the queue manager that is already used by WebSphere BI Connect. We then only need to add two new queues to this queue manager. In our environment, the queue manager runs on a separate machine, with host name wbicdata. On this machine, use the MQ command interface to define two new queues:

- RosettaNet_IN
- RosettaNet_OUT

Figure 21-2 shows a transcript of an interactive MQ command session using the `runmqsc` command to define these two queues.
21.3.2 Setting up JMS environment

We use JMS Integration in WebSphere BI Connect to pass RosettaNet Service Content and associated metadata between RFHUTIL and WebSphere BI Connect. Therefore, we need to configure the JMS layer in IBM WebSphere MQ version 5.3 to map the MQ queues to JMS queues.

In 15.2.2, “Enabling JMS” on page 427, we describe how to enable JMS on the machine that hosts the WebSphere BI Connect server. We continue to use this same JMS environment and simply add another queue connection factory and two queue destinations to the existing Java Naming and Directory Interface (JNDI) configuration.

1. Open a command window and change to the directory C:\WMQ\Java\bin.

2. Execute the program JMSAdmin, which is used to define the mapping between MQ and JMS and to store that mapping in a JNDI provider.

3. Figure 21-3 shows the use of JMSAdmin to create this mapping. Change to the existing context WBIC_JMS. This is the context used in Chapter 15, “Integration with WebSphere Data Interchange” on page 419.

4. Use the dis ctx command to see the contents of this context. It currently holds objects related to the integration with WebSphere Data Interchange.

5. The next command adds the queue connection factory. The command refers to the existing server-connection channel java.channel, which is created as part of the MQ configuration for WebSphere BI Connect. The definition of the

---

```
C:\>runmqsc partner_e.bcg.queue.manager
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Starting MQSC for queue manager partner_e.bcg.queue.manager.

define qlocal('RosettaNet_IN')
   1 : define qlocal('RosettaNet_IN')
AMQ8006: WebSphere MQ queue created.
define qlocal('RosettaNet_OUT')
   2 : define qlocal('RosettaNet_OUT')
AMQ8006: WebSphere MQ queue created.
end
   3 : end
2 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```
queue connection factory also refers to the host name of the machine that hosts the queue manager. And it refers to the port number of the listener.

The value 9999 is the default value for queue managers created by following the instructions of WebSphere BI Connect. However, it can be any value. The creation of this queue manager is part of Chapter 5, “Implementing WebSphere BI Connect Enterprise in a Windows environment” on page 57.

6. Use the end command to stop the administration tool.

```plaintext
5648-C60, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2002. All Rights Reserved.
Starting WebSphere MQ classes for Java(tm) Message Service Administration

InitCtx> chg ctx(WBIC_JMS)

InitCtx/WBIC_JMS> dis ctx

Contents of InitCtx/WBIC_JMS
  .bindings       java.io.File
    a  EDI_IN       com.ibm.mq.jms.MQQueue
    a  EDI_OUT      com.ibm.mq.jms.MQQueue
    a  WBIC_QCF     com.ibm.mq.jms.MQQueueConnectionFactory
    a  TO.EDI_IN    com.ibm.mq.jms.MQQueue

5 Object(s)
  0 Context(s)
  5 Binding(s), 4 Administered

InitCtx/WBIC_JMS> def qcf(WBICHUB) qmgr(partner_e.bcg.queue.manager)
            tran(client) chan(java.channel) host(wbicdata) port(9999)

InitCtx/WBIC_JMS> def q(RosettaNet_IN) queue(RosettaNet_IN)

InitCtx/WBIC_JMS> def q(RosettaNet_OUT) queue(RosettaNet_OUT)

InitCtx/WBIC_JMS> end

Stopping WebSphere MQ classes for Java(tm) Message Service Administration
C:\WMQ\Java\bin>
```

Figure 21-3 Using JMSAdmin

The previous steps change the .bindings file in the C:\WMQ\Java\JNDI\WBIC_JMS directory. Notice that the use of a context results
in the use of directory within the root directory of the file-based JNDI provider. Also, the commands in Figure 21-3 are independent of the JNDI provider used.

**Note:** The .bindings file is a text file. However, we recommend that you do not edit this file manually. Always use the JMSAdmin tool to make any changes.

### 21.3.3 Adding DUNS identifiers to each profile

RosettaNet requires Data Universal Numbering System (DUNS) numbers. These numbers are stored in a partner profile as a business identifier. So far, we have been using free-form business identifiers. Now we must add DUNS numbers to the profile of Company E and to the profile of Company A on the server of Company E.

1. Log on as hubadmin to the console of WebSphere BI Connect of Company E. Select **Account Admin** → **Profiles** → **Community Participant**.
2. Click **Search**.
3. When you see the list of profiles, click the magnifying glass icon next to the profile of **Company E**.
4. When the profile of Company E is shown, click the **Edit** icon to edit the details of this profile.
5. In the Company E Profile window (Figure 21-4), complete these tasks:
   a. Under the Business ID section, click **New**.
   b. For Type, select **DUNS** and, under Identifier, enter the DUNS number. For Company E, we choose the number 999999999.
   c. Click **Save**.
Figure 21-4 Adding a new business identifier of type DUNS to the profile

6. Repeat the procedure starting with step 3 to add a DUNS number to the profile of Company A on the server of Company E. For Company A, we set the DUNS number to 111111111.

**Note:** You need to perform the same changes on the server of Company A. Remember that each trading partner has a local profile on his own server and a remote profile on the server of his partner. There is no automatic update or synchronization between the two profiles.
21.3.4 Setting up targets and gateways

In this section, you create new targets and gateways on the seller’s system (Company E). You need:

- A JMS target that points to the queue RosettaNet_OUT
- An HTTP target that defines the entry point for RosettaNet on the server of Company E
- A JMS gateway that points to the queue RosettaNet_IN
- An HTTP gateway that points to the HTTP entry port on the server of Company A

For the HTTP target and gateway, we can reuse the existing target and gateway. The HTTP target on Company E was called /bcgreceiver/companye/edi_in and is defined in Chapter 7, “Creating a basic B2B exchange” on page 121. Similarly, we defined HTTP gateway http://9.42.171.84:57080/bcgreceiver/companya/edi_in in the profile of Company A on the server of Company E. Refer again to Chapter 7, “Creating a basic B2B exchange” on page 121.

You can reuse the same target and gateway. Even though the name of these objects refer to electronic data interchange (EDI), you use them for any kind of document. Previously, we used these objects to transfer XML documents to Company X, for example. However, for clarity, we define new targets and gateways specific for the exchange of RosettaNet documents.

1. To create the targets, log on as the hubadmin of Company E. Select Hub Admin → Hub Configuration → Targets.
2. When you see the list of targets, click Create Target.
3. Create a JMSTarget and enter the following information in the fields as shown in Figure 21-5:
   a. Name: Company_E_RN_OUT
   b. Status: Enabled
   c. Transport: JMS
   d. Gateway Type: Production
   e. JMS Provider URL: file:///C:/WMQ/Java/JNDI/WBIC_JMS
   f. JMS Queue Name: RosettaNet_OUT
   g. JMS Factory Name: WBICHUB
   h. JNDI Factory Name: com.sun.jndi.fscontext.RefFSContextFactory
   i. Accept the defaults for all other fields.
4. In the same way, create an HTTP target and enter the following information in the specified fields:
   a. Name: Company_E_RN_IN
   b. Status: Enabled
   c. Transport: HTTP/S
   d. Gateway Type: Production
   e. URL: /bcgreceiver/companye/rn_in
   f. Accept the defaults for all other fields.
Figure 21-6 shows the list of targets that are now defined on Company E.

Figure 21-6   List of targets on the server of Company E

5. Open now the profile of Company E and select Gateways.

6. Create a JMS gateway and enter the following information in the specified fields as shown in Figure 21-7:

   a. Name: Company_E_RN_IN
   b. Status: Enabled
   c. Online/Offline: Online
   d. Transport: JMS
   e. Target URI: file://C:/WMQ/Java/JNDI/WBIC_JMS
   f. JMS Factory Name: WBICHUB
   g. JMSMessage Class: TextMessage
   h. JMS Queue Name: RosettaNet_IN
   i. JMS JNDI Factory Name: com.sun.jndi.fscontext.RefFSContextFactory
   j. Accept the defaults for all other fields.
7. Open the profile of Company A and select **Gateways**.
8. Create an HTTP gateway and enter the following information in the specified fields as shown in Figure 21-8:
   a. Name: Company_A_RN_OUT
   b. Status: Enabled
   c. Online/Offline: Online
   d. Transport: HTTP/1.1
   e. Target URI: http://9.42.171.84:57080/bcgreceiver/companya/rn_in
   f. Accept the defaults for all other fields.

Figure 21-8 Creating an HTTP gateway in the profile of Company A

21.3.5 Uploading RosettaNet packages

Now that you have configured the targets and gateways, upload the RosettaNet packages needed to complete the 3A4 scenario. For RosettaNet messaging, WebSphere BI Connect requires the RNIF packages for the version of RNIF used to send and receive the messages. For each PIP that WebSphere BI Connect supports, it requires the PIP’s two PIP document flow packages to be uploaded into the system.
Chapter 21. Implementing a RosettaNet solution

- Package_RNIF_V02.00.zip: Enables WebSphere BI Connect to trade RNIF 2.0 messages to and from trading partners
- Package_RNSC_1.0_RNIF_V02.00.zip: Enables WebSphere BI Connect to pass RNSC to and from the back end
- BCG_Package_RNIFV02.00_3A4V02.02.zip: Enables WebSphere BI Connect to trade 3A4 RosettaNet Business Messages to and from partners
- BCG_Package_RNSC1.0_RNIFV02.00_3A4V02.02.zip: Enables WebSphere BI Connect to pass 3A4 Service Content to and from the back end

Follow these steps to upload RosettaNet packages into the Company E system:

1. Create a directory called B2BIntegrate\RosettaNet within the WebSphere BI Connect installation directory.

2. Copy the four RosettaNet packages (see previous list) into the newly created directory. You can find the RosettaNet packages on the WebSphere BI Connect installation CD in the \B2BIntegrate\RosettaNet directory.

3. If you have not already done so, log on to the Company E system as hubadmin. Click Hub Admin → Hub Configuration → Document Flow Definition.

4. Click Upload/Download Packages as shown in Figure 21-9.

![Figure 21-9 Manage document flow definitions](image)
5. In the Upload/Download Packages window (Figure 21-10), complete the following information:
   a. Select **No** for WSDL Package.
   b. Click **Browse** and navigate to the directory where you copied the RosettaNet packages. Select the **Package_RNIF_V02.00.zip** file.
   c. Ignore the Web Service Public URL field.
   d. For Commit to database, select **Yes**.
   e. For Overwrite data, choose **No**.
   f. Click the **Upload** button.

![Figure 21-10   Importing a package into WebSphere BI Connect](image-url)
6. After the file is uploaded into the system, the status box displays a message that indicates success (Figure 21-11). Provide the file name of the second package and upload it as well. Repeat this step until you upload all four files.

![WebSphere Business Integration Connect Community Console](image)

**Figure 21-11  Package uploaded successfully**
Verifying that the packages are uploaded

After all of the packages are uploaded, you can verify that they are correct.


2. In the Manage Document Flow Definitions view, click the folder next to Package: Backend Integration and Package: RosettaNet. Expand both packages as far as you can go. You should see all four packages that you uploaded as shown in Figure 21-12.
21.3.6 Creating interactions for RosettaNet

Now that you have uploaded the packages and the document flow definitions for RosettaNet, create the interactions. For each PIP, there needs to be an interaction for sending messages and one for receiving messages.

1. If you have not already done so, log on to the Company E system as hubadmin. Configure an interaction for receiving PO request RNIF messages. Select Hub Admin \(\rightarrow\) Hub Configuration \(\rightarrow\) Document Flow Definition.

2. When you see the document flow definitions, select Manage Interactions.

3. In the search window, select Create Interaction.

4. In the Manage Interactions window (Figure 21-13), complete these steps:
   a. Expand the source Document Flow Definition tree. Click Package: RNIF (V02.00) \(\rightarrow\) Protocol: RosettaNet (V02.00) \(\rightarrow\) Document Flow: 3A4 (V02.02) Request Purchase Order \(\rightarrow\) Activity: Request Purchase Order. Then select Action: Purchase Order Request Action.

   b. Expand the target Document Flow Definition tree. Click Package: Backend Integration (1.0) \(\rightarrow\) Protocol: RNSC (1.0) \(\rightarrow\) Document Flow: 3A4 (V02.02) Request Purchase Order \(\rightarrow\) Activity: Request Purchase Order. Then select Action: Purchase Order Request Action.

   c. Skip the transformation map fields.

   d. In the Action field, select Bi-Directional Translation of RosettaNet and RosettaNet Service Content with Validation.

   e. Click Save to save the interaction.
Figure 21-13 Receiving RosettaNet R.O. Request Business Messages
5. Configure an interaction for sending P.O. Confirmation RNIF Messages as shown in Figure 21-14. Repeat the same steps as in the previous step.


c. In the Action field, select Bi-Directional Translation of RosettaNet and RosettaNet Service Content with Validation.

d. Click Save to save the interaction.
You have now configured the hub of Company E to process purchase orders and purchase order confirmation. The hub expects to receive purchase orders from back-end applications with back-end integration packaging and to pass it to the target partner in RNIF packaging. Partners can send a purchase order confirmation in a RNIF packaging that is passed to back-end applications in the back-end integration packaging. Now you add these capabilities to the profiles of Company A and Company E on the hub of Company E.

21.3.7 Configuring the B2B capabilities

To configure the B2B capabilities of Company E, you can either log on as the hubadmin or as the admin user of Company E. Again we use the hubadmin user, who also configures the profiles of the partners.

1. Select Account Admin →Profiles →Community Participant.
2. Click Search.
3. Click the magnifying glass icon next to Company E.
5. In the B2B Capabilities window (Figure 21-15) of Company E, complete these steps:
   a. Enable the source and target for Package: Backend Integration (1.0) and Package: RNIF (V02.00).
   b. Expand Package: Backend Integration and enable the Source and Target for Protocol: RNSC (1.0).
   c. Expand Protocol: RNSC (1.0) and enable the Source and Target for Document Flow: 3A4 (V02.02) Request Purchase Order. This automatically enables all child entries under the document flow.
   d. Expand Package RNIF (V02.00) and enable the Source and Target for Protocol: RosettaNet (V02.00).
   e. Expand Protocol: RosettaNet (V02.00) and enable the Source and Target for Document Flow: 3A4 (V02.02) Request Purchase Order. This automatically enables all child entries under the document flow.
Figure 21-15  Company E’s B2B capabilities
6. Repeat this for Company A’s profile on the server of Company E. Figure 21-16 shows the B2B capabilities view for Company A.

You have now added the capabilities for RosettaNet to each partner’s profile. Next, you create participant connections.

21.3.8 Configuring the participant connections

You can create participant connections as hubadmin or as admin user of Company E. While logged on as hubadmin, perform these steps:

1. Click **Account Admin → Participant Connections**.
2. Select **Company E** as the source and **Company A** in as target. Click **Search**.
3. As shown in Figure 21-17, you see two inactive interactions that are related to RosettaNet. Activate the Backend Integration (Confirmation) to RNIF (Confirmation) and the RNIF (Request) to Backend Integration (Request) interactions.

a. Click the **Activate** button for both interactions.

b. Click **Gateways** for both interactions.

c. Change the source gateway to **Company_E_RN_IN** and the target gateway to **Company_A_RN_OUT**.

d. Click **Save** to persist the changes.

---

*Figure 21-17  Seller's system: Company E to Company A interaction*
In the same way, activate the reverse connection so that you can receive RosettaNet documents from Company A.

1. Choose **Company A** as the source and **Company E** in as the target. Click **Search**.

2. As shown in Figure 21-18, you see two inactive interactions that are related to RosettaNet. Activate the Backend Integration (Confirmation) to RNIF (Confirmation) and the RNIF (Request) to Backend Integration (Request) interactions.
   
a. Click the **Activate** button for both interactions.
   b. Click **Gateways** for both interactions.
   c. Change the source gateway to **Company_A_RN_OUT** and the target gateway to **Company_E_RN_IN**.
   d. Click **Save** to persist the changes.

---

### Manage Connections

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company A</td>
<td>Company E</td>
</tr>
</tbody>
</table>

---

**Figure 21-18** Seller’s system: Company A to Company E interaction
21.3.9 Providing failure notification

If a failure occurs during the processing of a PIP message, WebSphere BI Connect uses the 0A1 PIP as the mechanism to broadcast the failure to the participant or back-end system that sent the message. For example, a back-end system initiates a 3A4 PIP. WebSphere BI Connect processes the RNSC message and sends a RosettaNet message to a participant. WebSphere BI Connect waits for the response to that RosettaNet message until the waiting time reaches the time-out limit. When this occurs, WebSphere BI Connect creates a 0A1 PIP and sends it to the participant. The 0A1 PIP identifies the exception condition so that the participant can compensate for the failure of the 3A4 PIP.

To provide failure notification, follow these steps:

1. Upload a 0A1 package.
2. Create a PIP connection to the participant using this package.
3. Change the RosettaNet contact information with the 0A1 PIP. To do so, edit the BCG.Properties file, located in the <install_root>/router/was/wbic/config directory. These fields populate the contact information within the 0A1 PIP. Fax is optional (value can be empty), but the other attributes are required.
   - bcg.0A1.fromContactName
   - bcg.0A1.fromEMailAddr
   - bcg.0A1.fromPhoneNbr
   - bcg.0A1.fromFaxNbr
   The phone numbers are limited to 30 bytes in length. The other fields are unlimited in length.
4. After you change this file, restart the router.

You can also provide the values for these attributes during the installation of the software. However, it is possible to change these attributes by editing this properties file.

For the purpose of this scenario, we do not set up the 0A1 PIP at this time. If you want to set it up, you can use the same instructions for uploading packages, creating interactions, setting B2B capabilities, and configuring participant connections to enable the PIP. The process is exactly the same as outlined in this section for the 3A4 PIP.

21.4 Configuration tasks for hubadmin of Company A

This section explains the steps to complete the setup of RosettaNet in WebSphere BI Connect Advanced of Company A. The hubadmin must upload of RosettaNet packages because the hub administrator is the only user with rights
to upload packages into the system. The admin and the participant users can perform the remaining tasks much in the same way as demonstrated in Chapter 7, “Creating a basic B2B exchange” on page 121. For the purpose of this scenario, we perform all of the RosettaNet configuration tasks on Company A’s system as the hubadmin user.

Most tasks are not different for Company A as they were for Company E. We repeat the tasks in this section for the sake of completeness.

### 21.4.1 Setting up MQ queues

Since we do not perform any integration with back-end systems, we can simply use the queue manager that is already used by WebSphere BI Connect. Then we only need to add two new queues to this queue manager. In this environment, the queue manager runs on the same machine as the WebSphere BI Connect server. On this machine, use the MQ command interface to define two new queues:

- RosettaNet_IN
- RosettaNet_OUT

Figure 21-19 shows a transcript of an interactive MQ command session using `runmqsc` to define these two queues.

```
# su - mqm
$ runmqsc partner_a.bcg.queue.manager
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Starting MQSC for queue manager partner_a.bcg.queue.manager.

define qlocal('RosettaNet_IN')
1 : define qlocal('RosettaNet_IN')
AMQ8006: WebSphere MQ queue created.
define qlocal('RosettaNet_OUT')
2 : define qlocal('RosettaNet_OUT')
AMQ8006: WebSphere MQ queue created.
end
3 : end
2 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
$
```

*Figure 21-19  Using runmqsc to define queues*
21.4.2 Setting up the JMS environment

Since we use JMS integration in WebSphere BI Connect to pass RosettaNet Service Content and associated metadata to WebSphere BI Connect, we must configure the JMS layer in IBM WebSphere MQ version 5.3 to map the MQ queues to JMS queues.

WebSphere BI Connect uses the JMS API to interact with WebSphere MQ. The JMS API is part of the Java 2 Platform, Enterprise Edition (J2EE) specification. It is a layer developed on top of the standard WebSphere MQ API. We must also set up the mapping between JMS objects and WebSphere MQ objects.

The mapping is retrieved by the JMS API using the JNDI API. JNDI providers can be anything, for example Lightweight Directory Access Protocol (LDAP) servers or naming services provided by other WebSphere Application Server instances in your network or a simple file. The actual mapping commands are independent of the JNDI provider.

WebSphere MQ provides the JMSAdmin tool to store the links between a JMS object and a WebSphere MQ in a JNDI directory. The choice of JNDI provider is set up in the configuration file JMSAdmin.config, which you can find in the Java\bin directory in the installation folder of WebSphere MQ. You need to perform the mapping on the machine where the JMS application is running, and not on the machine that runs the actual JMS server (or queue manager).

Follow these steps:

1. On the WebSphere BI Connect machine, open the JMSAdmin.config file in the /usr/lpp/mqm/java/bin folder in a text editor.

2. Uncomment the INITIAL_CONTEXT_FACTORY setting for the JNDI provider of your choice. In Example 21-1 it is set for the file-based JNDI provider. Each JNDI provider needs to be addressed in a different way, which is expressed via the parameter PROVIDER_URL.

3. Update the file-based URL to point to a valid directory on the file system, for example /usr/lpp/mqm/java/JNDI.

Note: Make sure that only one setting of each parameter is uncommented. If you want to use multiple providers, you need to create multiple configuration files and pass the name of the configuration file as a parameter when running the JMSAdmin tool.
Example 21-1  JMSAdmin configuration file

```java
# The following line specifies which JNDI service provider is in use.
# It currently indicates an LDAP service provider. If a different
# service provider is used, this line should be commented out and the
# appropriate one should be uncommented.
#
#INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
#INITIAL_CONTEXT_FACTORY=com.ibm.websphere.naming.WsnInitialContextFactory
#
# The following line specifies the URL of the service provider's initial
# context. It currently refers to an LDAP root context. 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:/usr/lpp/mqm/java/JNDI
#PROVIDER_URL=iiop://localhost/
```

4. The /usr/lpp/mqm/java/JNDI directory does not exist by default. You must
   create it and make sure that it is owned by the correct user ID and group ID.
   Use the following commands to do this:

   ```bash
   cd /usr/lpp/mqm/java
   mkdir JNDI
   chown mqm:mqm JNDI
   
   5. You may also need to update the profile of the mqm user to update or set the
      CLASSPATH and the PATH environment variables.

      a. Open the file .profile in the home directory of the user mqm (for example
         /var/mqm).

      b. Make the required changes. Example 21-2 shows sample declarations for
         PATH and CLASSPATH.
Example 21-2  Updated profile for mqm user

PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:/usr/java131/jre/bin:/usr/java131/bin:


export PATH

export CLASSPATH

if [ -s "$MAIL" ] # This is at Shell startup. In normal
then echo "$MAILMSG" # operation, the Shell checks
fi # periodically.

6. Activate the new profile without logging off and on again. Enter the following command in the home directory of the user mqm:

```
./.profile
```

Notice that this is a dot, followed by a space, followed by another dot, followed by the slash character and another dot and the word “profile”.

You can now use the JMSAdmin tool to create the mapping between JMS objects and MQ objects.

1. Start JMSAdmin in a command window with /usr/lpp/mqm/java/bin as the current directory. We need to create a mapping between a queue connection factory object and a queue manager. We also need two mappings between the JMS queues for inbound and outbound messages and the MQ queues that we defined in 21.4.1, “Setting up MQ queues” on page 734.

2. Figure 21-20 shows the use of JMSAdmin to create this mapping.

   a. Add a new context to store the JMS objects for use by WebSphere BI Connect using the `def ctx(WBIC_JMS)` command.

   b. Change to this new context WBIC_JMS using the `chg ctx(WBIC_JMS)` command.

   c. The next command adds the queue connection factory. It refers to the existing queue manager partner_a.bcg.queue manager. The creation of this queue manager is part of Chapter 6, “Implementing WebSphere BI Connect Advanced for AIX” on page 95.
d. The two other commands create a binding between the JMS queues RosettaNet_IN and RosettaNet_OUT and the two MQ queues with the same name.

3. Use the **end** command to stop the administration tool.

These steps change the .bindings file in the /usr/lpp/mqm/java/JNDI/WBIC_JMS directory. Notice that the use of a context results in the use of directory within the root directory of the file-based JNDI provider.

**Note:** The .bindings file is a text file. We recommend that you do not try to edit this file manually. Always use the tool JMSAdmin to make any changes.

Also, the commands in Figure 21-20 are independent of what JNDI provider is used.

```
$ JMSAdmin
5648-C60, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2002. All Rights Reserved.
Starting WebSphere MQ classes for Java(tm) Message Service Administration
InitCtx> def ctx(WBIC_JMS)
InitCtx> chg ctx(WBIC_JMS)
InitCtx/WBIC_JMS> def qcf(WBIC_CPYA) qmanager(partner_a.bcg.queue.manager)
InitCtx/WBIC_JMS> def q(RosettaNet_IN) queue(RosettaNet_IN)
InitCtx/WBIC_JMS> def q(RosettaNet_OUT) queue(RosettaNet_OUT)
InitCtx/WBIC_JMS> end
Stopping WebSphere MQ classes for Java(tm) Message Service Administration
$
```

*Figure 21-20  Using JMSAdmin*
21.4.3 Adding the DUNS IDs to each profile

The exchange of the 3A4 document assumes that we are using DUNS numbers as our business IDs. To facilitate this, we add a second ID for the Company E and Company A profiles.

1. Edit Company A’s profile. Add a Business ID of type DUNS and enter 111111111 as the DUNS Number.
2. Edit Company E’s profile. Add a Business ID of type DUNS and enter 999999999 as the DUNS Number.

The process to add DUNS numbers to a profile in WebSphere BI Connect is explained in 21.3.3, “Adding DUNS identifiers to each profile” on page 714. This time, perform this task on the server of Company A and add DUNS numbers to the profiles of Company A and Company E on the server of Company A.

21.4.4 Setting up targets and gateways

This section explains how to create new targets and gateways on the buyer’s system (Company A). You have a separate channel for RosettaNet documents.

Note: Remember, that when you are logged in as hubadmin and configuring gateways for your participants, you must click Account Admin → Profiles → Search. Select the participant you want to create the gateway for and then click the Gateways menu item.

1. Log on to the Company A system as hubadmin. Select Hub Admin → Hub Configuration → Targets.
2. Create a JMS target and enter the following information in the specified fields:
   a. Name: Company_A_RN_OUT
   b. Status: Enabled
   c. Transport: JMS
   d. Gateway Type: Production
   e. JMS Provider URL: file://usr/lpp/mqm/java/JNDI/WBIC_JMS
   f. JMS Queue Name: RosettaNet_OUT
   g. JMS Factory Name: WBIC_CPYA
   h. JNDI Factory Name: com.sun.jndi.fscontext.RefFSContextFactory
   i. Accept the defaults for all other fields.
3. Create an HTTP target and enter the following information in the specified fields:
   
a. Name: Company_A_RN_IN
b. Status: Enabled
c. Transport: HTTP/S
d. Gateway Type: Production
e. URI: /bcgreceiver/companya/rn_in
f. Accept the defaults for all other fields.

Figure 21-21 shows the list of targets that are currently defined on the server of Company A.

![Image of WebSphere Business Integration Connect Community Console](image)

Figure 21-21  List of targets on the server of Company A
4. Open the profile of Company A and select **Gateways**.

5. Create a JMS Gateway for Company A’s profile. Enter the following information in the specified fields:

   a. Name: **Company_A_RN_IN**
   b. Status: **Enabled**
   c. Online/Offline: **Online**
   d. Transport: **JMS**
   e. Target URI: file:///usr/lpp/mqm/java/JNDI/WBIC_JMS
   f. JMS Factory Name: **WBIC_CPYA**
   g. JMS Message Class: **TextMessage**
   h. JMS Queue Name: **RosettaNet_IN**
   i. JMS JNDI Factory Name: com.sun.jndi.fscontext.RefFSContextFactory
   j. Accept the defaults for all other fields.

   Figure 21-22 shows the currently defined gateways of Company A on the server of Company A.

---

**Figure 21-22   List of gateways in the profile of Company A**
6. Open the profile of Company E on the server of Company A and select **Gateways**.

7. Create an HTTP gateway in Company E’s profile and enter the following information in the specified fields:
   
a. **Name:** Company_E_RN_OUT  
b. **Status:** Enabled  
c. **Online/Offline:** Online  
d. **Transport:** HTTP/1.1  
e. **Target URI:** http://wbichub:57080/bcgreceiver/companye/rn_in  
f. Accept the defaults for all other fields.

Figure 21-23 shows the list of defined gateways in the profile of Company E on the server of Company A.

![Figure 21-23   List of gateways in profile of Company E on server of Company A](image)

### 21.4.5 Uploading the RosettaNet packages

Now that you have configured the targets and gateways, you must upload the RosettaNet packages needed to complete the 3A4 scenario. For RosettaNet messaging, WebSphere BI Connect requires the RNIF packages for the version of RNIF used to send and receive the messages. For each PIP that WebSphere BI Connect supports, it requires the PIP’s two PIP document flow packages to be uploaded into the system.

- Package_RNIF_V02.00.zip: Enables WebSphere BI Connect to trade RNIF 2.0 messages to/from trading partners
- Package_RNSC_1.0_RNIF_V02.00.zip: Enables WebSphere BI Connect to pass RNSC to and from the back end
- **BCG_Package_RNIFV02.00_3A4V02.02.zip**: Enables WebSphere BI Connect to trade 3A4 RosettaNet Business Messages to and from partners
- **BCG_Package_RNSC1.0_RNIFV02.00_3A4V02.02.zip**: Enables WebSphere BI Connect to pass 3A4 Service Content to and from the back end

You can find RosettaNet packages on the WebSphere BI Connect installation CD in the ‘\B2BIntegrate\rosettanet’ directory. The process of uploading these packages is the same as before. For details, see 21.3.5, “Uploading RosettaNet packages” on page 720. You need to upload these packages in the correct order.

**Verifying the package uploads**

After you upload all of the packages, verify that they are correct.

1. Select **Hub Admin →Hub Configuration →Document Flow Definition**.

2. In the Manage Document Flow Definitions View (Figure 21-24), click the folder next to **Package: Backend Integration** and **Package: RosettaNet**. Expand both packages as far as you can go. You should see all four packages that you uploaded.

![Manage Document Flow Definitions](Figure 21-24 Verifying the RosettaNet packages)
21.4.6 Creating interactions for RosettaNet

You have now uploaded the packages and the document flow definitions for RosettaNet. Now create interactions. For each PIP, there needs to be one interaction for sending messages and one interaction for receiving messages. The process is exactly the same as described for Company E in 21.3.6, “Creating interactions for RosettaNet” on page 725.

Configure participant connections between the back end and the participant (Company E).

1. If you have not already done so, log on to the Company A system as hubadmin.

2. Configure an interaction for receiving P.O. Request RNIF messages.
   b. Click Manage Interactions.
   c. Click Create Interaction.
   d. In the Manage Interactions window (Figure 21-25), follow these steps:
      iii. Skip the transformation map fields.
      iv. In the Action field, select Bi-Directional Translation of RosettaNet and RosettaNet Service Content with Validation.
      v. Click the Save button to save the interaction.
3. Configure an interaction for sending P.O. Request RNIF Messages.
   a. Click **Hub Admin** → **Hub Configuration** → **Document Flow Definition**.
   b. Click **Manage Interactions**.
   c. Click **Create Interaction**.
   d. In the Manage Interactions window (Figure 21-26), complete these steps:
      i. Expand the source Document Flow Definition tree. Click **Package**: **Backend Integration (1.0)** → **Protocol**: **RNSC (1.0)** → **Document Flow**: **3A4 (V02.02) Request Purchase Order** → **Activity**: **Request Purchase Order**. Select **Action**: **Purchase Order Request Action**.
      ii. Expand the target Document Flow Definition tree. Click **Package**: **RNIF (V02.00)** → **Protocol**: **RosettaNet (V02.00)** → **Document Flow**: **3A4 (V02.02) Request Purchase Order** → **Activity**: **Request Purchase Order**. Select **Action**: **Purchase Order request Action**.
iii. Skip the transformation map fields

iv. In the Action field, select **Bi-Directional Translation of RosettaNet and RosettaNet Service Content with Validation**.

v. Click the **Save** button to save the interaction.

---

**Figure 21-26  Sending RosettaNet P.O. Request Business Messages**
21.4.7 Configuring B2B capabilities

To configure the B2B capabilities of Company A, you can either log on as the hubadmin or as the admin user of Company A. Again we use the hubadmin user, who configures also the profiles of the partners.

1. Click **Account Admin → Profiles**.
2. Click **Search**.
3. Click the magnifying glass icon next to **Company A**.
4. Click **B2B Capabilities**.
5. In the B2B Capabilities window for Company A, enable the appropriate source and target document flow definitions, as shown in Figure 21-27.

![Company A's B2B capabilities](image)

*Figure 21-27 Company A's B2B capabilities*
6. Repeat this for Company E’s profile on the server of Company A.
Figure 21-28 shows the B2B capabilities view for Company A.

![Image](image-url)

**Figure 21-28  Company E’s B2B capabilities**

### 21.4.8 Configuring participant connections

Now that you have added the capabilities for RosettaNet to each partner’s profile, you can create participant connections.

You can create the participant connections as the hubadmin or as admin user of Company E. While logged on as hubadmin, perform the following steps:

1. Click **Account Admin → Participant Connections**.
2. Choose **Company A** as the source and **Company E** in as the target and click **Search**.
3. In the Manage Connection window (Figure 21-29), you see two inactive interactions. Activate the Backend Integration (Request) to RNIF (Request) and the RNIF (Confirmation) to Backend Integration (Confirmation) interactions.

   a. Click **Activate** for both interactions.
   
   b. Click **Gateways** for both interactions.
   
   c. Change the source gateway to **Company_A_RN_IN** and the target gateway to **Company_E_RN_OUT**.
   
   d. Click **Save** to persist the changes.

---

**Figure 21-29**  Buyer’s system: Company A to Company E interaction
4. Select **Company E** as the source and **Company A** as the target. Click **Search**.

5. In the Manage Connections window (Figure 21-30), you see two inactive interactions related to RosettaNet.
   a. Click **Activate** for both interactions.
   b. Click **Gateways** for both interactions.
   c. Change the source gateway to **Company_E_RN_OUT** and the Target Gateway to **Company_A_RN_IN**.
   d. Click **Save** to persist the changes.

---

**Figure 21-30  Buyer's system: Company E to Company A interaction**
21.5 Validating the RosettaNet configuration

Now that we have configured both the seller’s and buyer’s systems to trade RosettaNet using the 3A4 PIP, we can trade Purchase Order Requests and Confirmations between the two instances of WebSphere BI Connect. In this section, we use the same WebSphere BI Connect instances that were used for all scenarios in this redbook. Therefore, digital certificates have been exchanged and uploaded, as discussed in Chapter 8, “Securing the B2B exchange” on page 189.

As stated in 21.2, “Trading scenario overview” on page 708, we use the RFHUTIL utility to simulate the back-end system. To do this, you must load a payload file into the utility and configure the appropriate tabs to post the file and required metadata to the JMS queues on each machine. To obtain these test files, refer to Appendix B, “Additional material” on page 773.

This scenario assumes that RFHUTIL and the test files are installed on and executed from each WebSphere BI Connect system. For instructions on how to use RFHUTIL, refer to the user documentation of this SupportPac.

- 3A4_Request_RNSC.xml
- 3A4_V02.02_Request.txt
- 3A4_Confirmation_RNSC.xml
- 3A4_V02.02_Request.txt

**Note:** Company A has installed WebSphere BI Connect on AIX, for which there is no version of RFHUTIL. However, you can use the MQ client version of RFHUTIL (called RFHUTILC) and run it on a Windows machine. The simplest way to set up an MQ client connection is to create an environment variable:

```
SET MQSERVER=java.channel/TCP/m106984f(9999)
```

The value `java.channel` is the name of a server connection channel that is always created on the queue manager used by WebSphere BI Connect. The last parameter is the host name and port of the server that hosts the queue manager.
21.5.1 Sending the P.O. request message from a buyer to seller

The first action of the 3A4 PIP is to send a P.O. request from the buyer to the seller. Use the following steps to configure RFHUtil and to send the request action to the seller.

1. If you have not already done so, launch all WebSphere BI Connect components (Console, Receiver, Document Manager) on Company E and Company A systems.

2. On Company A’s system, start RFHUTIL and open the 3A4_V02.02_Request.txt file.

3. In the Main tab (see Figure 21-31), complete these tasks:
   a. Select the queue manager used by WebSphere BI Connect, `partner_a.bcg.queue.manager`.
   b. Enter the name of the queue that is used as the JMS target, `RosettaNet_OUT`.

![RFHUtil Main tab](image)

*Figure 21-31  RFHUtil Main tab*
4. Click the **Data** tab (Figure 21-32) to verify that the file is loaded. Under Data Format, select the **XML** option to reformat the data as an XML document.

![Figure 21-32  RFHUtil Data tab](image)
5. Click the **MQMD** tab (Figure 21-33). Verify the following fields:
   a. MQ Message Format must indicate MQHRF2.
   b. Code Page should indicate 437.
   c. All other fields can have the default values.

![RFHUtil MQMD tab](image)
6. Click the **RFH** tab (Figure 21-34) and verify the following fields:

   a. Under the RFH V2 Fixed Data section, set the following selections.
      i. For Integer, select **PC**.
      ii. For Data Format, specify MQSTR.
      iii. For Code Page, specify 1208.
      iv. For CCSID, select **1208**.

   b. For Message Domain, type **jms_text**.

   c. For Msg Type, enter MQSTR.

   d. For V2 Folders, select **mcd, jms, and usr**.

   e. For RFH Type, select **Version 2**.

![Figure 21-34  RFHUtil RFH tab](image-url)
7. Click the **jms** tab (Figure 21-35) and verify the following fields:

   a. For JMS Message Type, select **text**.
   b. For Destination, type **queue:///RosettaNet_OUT**.
   c. For Delivery Mode, type 2.
8. The **usr** tab is where you place the metadata that WebSphere BI Connect requires for receiving files from the back end.
   
a. Open the **3A4_Request_Header.txt** file in a text editor.
b. Select all of the text in the file.
c. Copy the data and paste it into the Usr folder contents field as shown in Figure 21-36.

**Note:** If you intend to send more than one test message to the WebSphere BI Connect system, you must change the `<x_aux_system_msg_id>` element to a different number. Otherwise WebSphere BI Connect will reject the message as a duplicate.
For clarity, Figure 21-37 shows the usr folder contents in a text format. The text is slightly edited for readability.

```xml
<x_aux_transport_retry_count>1</x_aux_transport_retry_count>
<x_aux_msg_id>3A4PIP_Req_26Aug_2004</x_aux_msg_id>
<x_aux_payload_root_tag>Pip3A4PurchaseOrderRequest</x_aux_payload_root_tag>
<x_aux_production>Production</x_aux_production>
<x_aux_protocol>RNSC</x_aux_protocol>
<x_aux_sender_id>111111111</x_aux_sender_id>
<x_aux_process_version>V02.02</x_aux_process_version>
<x_aux_protocol_version>1.0</x_aux_protocol_version>
<x_aux_process_instance_id>3A4PIP_014</x_aux_process_instance_id>
<x_aux_system_msg_id>3A4PIP_014</x_aux_system_msg_id>
<x_aux_process_type>3A4</x_aux_process_type>
<x_aux_receiver_id>999999999</x_aux_receiver_id>
```

Figure 21-37  Contents of the usr folder of a RosettaNet message

9. Return to the Main tab. Click **Write Q** to send the file to the queue **RosettaNet_OUT**.

### 21.5.2 Sending the confirmation message from seller to buyer

The second action of the 3A4 PIP is to send a P.O. confirmation from the seller to the buyer. The steps required to configure RFHUtil and send the confirmation are the same as in 21.5.1, “Sending the P.O. request message from a buyer to seller” on page 752. An exception applies to the payload file that you load into RFHUTIL and the metadata that you paste into the usr tab.

Follow the steps in the previous section for the confirmation. However, use the 3A4_Confirmation_RNSC.xml file and copy and paste from the 3A4_Confirmation_Header.txt as metadata in the usr folder.

### 21.5.3 Viewing RosettaNet transactions

You can use the Document Viewer in the Community Console to view individual documents that make up a process. The RosettaNet Viewer displays the choreography of documents that make up the entire business process.

1. Log on to Company A’s Community Console as hubadmin.
2. Select **Viewers →RosettaNet Viewer** to inspect the transaction result.
3. Figure 21-38 shows the completed transaction as a result of sending the XML file to Company E and receiving the response from Company E. Click the magnifying glass icon in front of the transaction to inspect the details of this transaction.
4. Figure 21-39 shows the incoming document as it is sent by RFHUTIL. Under it, you see the packaged document, which is what WebSphere BI Connect sends to Company E. The third document refers to the acknowledgement sent back by Company E. At the top, you see the type of document (3A4) and the instance ID.

![Figure 21-39  Details of a RosettaNet transaction](image)

a. Select the first document icon.

You now see the raw document as shown in Figure 21-40. You can see the transport headers, that were set by the JMS target. Under them, you see the actual document that was loaded by RFHUTIL.
b. Click the second document icon in Figure 21-39.

Now you see the message that was sent by WebSphere BI Connect. This message is packaged by WebSphere BI Connect so that it is a valid RosettaNet message. This document, as shown in Figure 21-41, displays...
the transport header, which is created by the HTTP gateway. Under this header is the translated document, which has multiple parts. You also see the preamble and the delivery header of this document.

Figure 21-41 Outgoing RosettaNet document (Part 1 of 2)
Scroll down in Figure 21-41 until you find the Service Header, which is shown in Figure 21-42. Figure 21-42 also shows the start of the RosettaNet Service Content, which is the actual payload of the message.
5. Now look at the RosettaNet Viewer on the server of Company E. Figure 21-43 shows the list of RosettaNet transactions on the server of Company E. Click the magnifying glass icon in front of the transaction.

![Figure 21-43  Transaction list in the RosettaNet viewer](image)

Now you can inspect the details, which are shown in Figure 21-44. The top two documents in Figure 21-44 are basically the same as shown earlier. The first one is the document that was received on the HTTP target of Company E. This is the packaged document. Under it, you see the document that is passed to the back-end applications by placing it in the JMS gateway queue RosettaNet_IN.

The two other documents represent the acknowledgement that is sent by Company E to Company A.
Figure 21-44  Details of a RosettaNet transaction

Figure 21-45 shows such an acknowledgment.

Figure 21-45  RosettaNet acknowledgment
Figure 21-46 and Figure 21-47 show the acknowledgment that is packaged by WebSphere BI Connect and sent to Company A. You can again see the preamble and the delivery header (in Figure 21-46).
Figure 21-47 shows a portion of the service header in the acknowledgment. At the bottom of Figure 21-47, you can see the digital signature that is added by WebSphere BI Connect.

![XML code snippet for the service header and digital signature](image-url)
Hardware and software configuration

This appendix provides details about the hardware and software configuration used to build a WebSphere BI Connect solution for this redbook.
Configuration for the hub machine

The hardware configuration used for running WebSphere BI Connect Enterprise in our lab consisted of:

- ThinkCentre™ M50 IBM PC (model 8189-F6U)
- Pentium® 4 processor running at 3.0 GHz
- 3 GB memory
- 40 GB hard disk

The software configuration used for this part of the solution, consisted of:

- Windows 2000 Server with Service Pack 4
- IBM DB2 Universal Database (UDB) V8.1.4 (only client component is used)
- IBM WebSphere MQ V5.3 with CSD7 (only client component is used)
- IBM WebSphere BI Connect Enterprise V4.2.2 and fix pack 3

Configuration for the data machine

This machine was used as the database server and MQ server for the hub machine. It was connected to it through a 100 Mb Ethernet network.

The hardware configuration used for this machine entailed:

- IBM @server xSeries® 230 (model 8658-61Y)
- Pentium III running at 1 GHz
- 2 GB memory
- Four SCSI hard disks of 36 GB

The software configuration used for this part of the solution consisted of:

- Windows 2000 Server with Service Pack 4
- IBM DB2 UDB V8.1.4
- IBM WebSphere MQ V5.3 with CSD7
- WebSphere MQ SupportPac MA0C (MQ Publish/Subscribe)

Configuration for the EDI translation machine

This machine was used to translate electronic data interchange (EDI) and XML documents and connected to the data machine via WebSphere MQ over a 100 Mb Ethernet network.
The hardware configuration used for this machine consisted of:

- IBM NetVista™ PC (model 6792-MHU)
- Pentium 4 running at 1.8 GHz
- 1.5 GB memory
- 40 GB hard disk (7200 RPM)

The software configuration used for this part of the solution consisted of:

- Windows 2000 Server with Service Pack 4
- IBM DB2 UDB V8.1.4
- IBM WebSphere MQ V5.3 with CSD7
- IBM WebSphere Data Interchange V3.2 Server with CSD12
- IBM WebSphere Data Interchange V3.2 Client with CSD8

**Configuration for the UNIX machine of Company A**

This machine was used to run WebSphere BI Connect for Company A. It was connected to the other machines over a 100 Mb Ethernet network.

The hardware configuration used for Company A consisted of:

- IBM @server pSeries® 630 (model 7028-6E4)
- POWER4™+ processor running at 1.20 GHz
- 16 GB memory
- Two SCSI hard disks of 36 GB

The software configuration used for this part of the solution consisted of:

- AIX 5L™ version 5.2
- IBM DB2 UDB V8.1.2
- IBM WebSphere MQ V5.3 with CSD7
- IBM WebSphere BI Connect Advanced V4.2.2 and fix pack 3

**Configuration for the Windows machine of Company X**

This machine was used to run WebSphere BI Connect Express for company X. It was connected to the other machines over a 100 Mb Ethernet network.

The hardware configuration used for this machine consisted of:

- IBM NetVista PC (model 6792-MHU)
- Pentium 4 running at 1.8 GHz
- 1.5 GB memory
- 40 GB hard disk (7200 RPM)
The software configuration used for this part of the solution consisted of:

- Windows 2000 Server with Service Pack 4
- WebSphere BI Connect Express V4.2.1

**Configuration for the Windows machine of Company F**

This machine was used to run the FTP server for company F. It was connected to the other machines over a 100 Mb Ethernet network.

The hardware configuration used for this machine consisted of:

- IBM NetVista PC (model 6792-MHU)
- Pentium IV running at 1.8 GHz
- 1.5 GB memory
- 40 GB hard disk (7200 RPM)

The software configuration used for this part of the solution consisted of:

- Windows 2000 Server with Service Pack 4
- WFTPD Pro V3.2.1 R3
Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246355

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the Additional materials and open the directory that corresponds with the redbook form number, SG246355.
**Using the Web material**

The additional Web material that accompanies this redbook is packaged as a zip file. After unzipping, you have four different folders containing several sample files that were used in the scenarios described in this redbook.

<table>
<thead>
<tr>
<th>Folder name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTDs</td>
<td>Two document type definitions (DTDs): one for purchase orders and one for inventory inquiry</td>
</tr>
<tr>
<td>Maps</td>
<td>Five import files of WebSphere Data Interchange containing maps and related definitions</td>
</tr>
<tr>
<td>Sample XML</td>
<td>XML documents that can be sent by Company E and Company X</td>
</tr>
<tr>
<td>Sample EDI</td>
<td>EDI documents that can be sent by Company E and Company A</td>
</tr>
<tr>
<td>User Exits</td>
<td>A collection of JAR files that contain the user exit classes discussed in Part 4 and a number of prerequisite JAR files</td>
</tr>
<tr>
<td>RosettaNet</td>
<td>Test files for input to RFHUTIL to validate the RosettaNet 3A4 PIP implementation in WebSphere BI Connect</td>
</tr>
</tbody>
</table>

**System requirements for downloading and using the Web material**

To implement any of the described scenarios, you must have access to at least two machines. The following system configuration is recommended:

- **Hard disk space:** 40 GB
- **Operating System:** Windows 2000 with Service Pack 4
- **Processor:** Pentium class processor running at 2 GHz or faster
- **Memory:** 2 GB or more
## Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>access control list</td>
</tr>
<tr>
<td>AIX</td>
<td>Advanced Interactive Executive</td>
</tr>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>API</td>
<td>application programming interface</td>
</tr>
<tr>
<td>BGP</td>
<td>Border Gateway Protocol</td>
</tr>
<tr>
<td>BI</td>
<td>Business Integration</td>
</tr>
<tr>
<td>BPI</td>
<td>Business Process Integration</td>
</tr>
<tr>
<td>BPM</td>
<td>business process management</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>COBOL</td>
<td>Common Business Oriented Language</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSD</td>
<td>Corrective Service Diskette</td>
</tr>
<tr>
<td>DCOM</td>
<td>Distributed Component Object Model</td>
</tr>
<tr>
<td>DER</td>
<td>Distinguished Encoding Rules</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Server</td>
</tr>
<tr>
<td>DTD</td>
<td>document type definition</td>
</tr>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration</td>
</tr>
<tr>
<td>EDI</td>
<td>electronic document interchange</td>
</tr>
<tr>
<td>EDIFACT</td>
<td>Electronic Data Interchange For Administration, Commerce &amp; Transport</td>
</tr>
<tr>
<td>EDI-INT</td>
<td>Electronic Data Interchange-Internet Integration</td>
</tr>
<tr>
<td>ERP</td>
<td>enterprise resource planning</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GML</td>
<td>Generalized Markup Language</td>
</tr>
<tr>
<td>GNU</td>
<td>GNU Not UNIX</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>HACMP™</td>
<td>High Availability Cluster Multiprocessing</td>
</tr>
<tr>
<td>HIPAA</td>
<td>Healthcare Information Portability and Accountability Act</td>
</tr>
<tr>
<td>HKLM</td>
<td>HKEY_LOCAL_MACHINE</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Mark-up Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Mail Access Protocol</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JNDI</td>
<td>Java Naming and Directory Service</td>
</tr>
<tr>
<td>JVM</td>
<td>Java virtual machine</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>MDN</td>
<td>Message Disposition Notification</td>
</tr>
<tr>
<td>MIME</td>
<td>Multi-purpose Internet Mail Extensions</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>MP</td>
<td>Multiplatform</td>
</tr>
<tr>
<td>MSCS</td>
<td>Microsoft Cluster Server</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organization for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>ODBC</td>
<td>Open Database Connectivity</td>
</tr>
<tr>
<td>ODETTTE</td>
<td>Organization for Data Exchange through Teletransmission in Europe</td>
</tr>
<tr>
<td>PKI</td>
<td>Private Key Infrastructure</td>
</tr>
<tr>
<td>RFH</td>
<td>Rules and Formats Header</td>
</tr>
<tr>
<td>RPM</td>
<td>rounds per minute</td>
</tr>
<tr>
<td>SCSI</td>
<td>Small Computer System Interface</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SGML</td>
<td>Standard Generalized Markup Language</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TPA</td>
<td>Trading Partner Agreement</td>
</tr>
<tr>
<td>UCS</td>
<td>Universal Multi-Octet Coded Character Set</td>
</tr>
<tr>
<td>UDB</td>
<td>Universal Database</td>
</tr>
<tr>
<td>UDDI</td>
<td>Universal Descriptions, Discovery and Integration</td>
</tr>
<tr>
<td>UN/CEFACT</td>
<td>United Nations Center for Trade Facilitation and Electronic Business</td>
</tr>
<tr>
<td>UNTDI</td>
<td>United Nations guidelines on Trade Data Interchange</td>
</tr>
<tr>
<td>URI</td>
<td>Universal Resource</td>
</tr>
<tr>
<td>URL</td>
<td>Universal Resource Locator</td>
</tr>
<tr>
<td>VAN</td>
<td>value-added network</td>
</tr>
<tr>
<td>VDA/SEDAS</td>
<td>Verband Deutscher Automobilhersteller / Standardisiertes Einheitliches DatenAustauschSystem</td>
</tr>
<tr>
<td>VICS</td>
<td>Voluntary Inter-industry Communications Standards</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 778. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Business Integration Server Express: The Express Route to Business Integration*, SG24-6353
- *WebSphere Application Server - Express V5.0.2 Developer Handbook*, SG24-6555
- *Implementing EDI Solutions*, SG24-6906
- *WebSphere Studio Application Developer Version 5 Programming Guide*, SG24-6957
- *Implementing and Administering WebSphere Business Integration Server V4.2.2*, SG24-7006
- *WebSphere Data Interchange Installation and Configuration*, REDP-3600

Online resources

These Web sites and URLs are also relevant as further information sources:

- WebSphere MQ Family SupportPacs
- WebSphere MQ Family support information
- WebSphere MQ Family Library
- DB2 Information Center
How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Index

A
access control list
   See ACL
access rights 310
ACL
   FTP server 310
      what is 26, 775
action
   provided by WebSphere BI Connect 496
      See also variable workflow
activate participant connections 165
ANSI X12
   what is 25
any-to-any transformation 379
application interaction 505
application trading partner 411
applies() method 630, 633, 639, 642, 644
AS2
   AS packaging failure 354
      identifier
         in WebSphere BI Connect Express 246
      setting MDN attributes 168
      viewing a packaged document 348
AS2 connection parameters
   identifier 156
   requested MDN 168
   update for digital encryption 226
   updating for encryption 194, 216
AS2 Viewer
   in Express 276
   in WebSphere BI Connect 347
assign gateway to connection 167
asynchronous request 539
attachment 704
authentication failure 360

B
B2B capabilities
   enabling 155, 160, 270
      in WebSphere BI Connect Express 246
B2B exchange 249
B2B gateway solution 383
B2C (business-to-consumer) 5
B2M2B (business-to-marketplace-to-business) 7
basic exchange 47
batch enveloper/de-enveloper 378
batch mode 387
BCG header 549
BCGException 516
BCGReceiverUtil class 551
black box testing 513
blocking mode 539
built-in profile 38
business directory 701
business message security 704
BusinessProcessFactoryInterface 631
BusinessProcessInterface 632
business-to-business (B2B) 5
   features and standards 39
   managing the infrastructure 52
   requirements 22
business-to-business integration (B2Bi) 6
business-to-consumer (B2C) 5
business-to-marketplace-to-business (B2M2B) 7

C
CA certificate 321
certificate 28
certificate authority 194, 221, 322
   upload public certificate 223
certificate errors 357
certificate type 224
circular log 69, 102
command
   bind 391
   connect to database 80
crtmqm 102
db2level 61, 96
db2set 64
FTP commands 282, 334
JMSAdmin 429
lslpp 97
mqver 59, 98
runmqsc 72, 103, 712
setmqcap 103
command chaining 409
configure for stored procedures
  AIX  101
  Windows  64
documentation  101
environment variables  64, 101
requirements for stored procedures
  AIX  100
  Windows  62
verify level
  AIX  96
  Windows  62
decryption  190
default gateway  151
default password
  of WebSphere BI Connect  89
  of WebSphere BI Connect Express  242
default user
  of WebSphere BI Connect  89
  of WebSphere BI Connect Express  242
define
  group
    AIX  102
    Windows  67
  user ID
    AIX  102
    FTP server  308
    Windows  66
delivery failure  341, 344
delivery header  702, 704
development environment  386
development tool  511
digital signature  28, 190, 219
  certificate authority  225
  enabling  222
  obtaining certificates  194
requirements for WebSphere BI Connect  221
  RosettaNet  702
  solving problems  358
  unable to verify  363
  update participant connections  226
  validate  231
  validation requirements  364
  verifying that they are enabled  231
  versus encryption  190
directory structure
  for FTP server  303
  for FTP target  286
document flow  132
document flow definition  132
create  259–260
  document flow type  261
  protocol type  259
  link to packaging  259
  upload package  721
document type definition  29
document viewer
  EDI  346
  FTP  299, 315
documentation
  DB2  101
  WebSphere BI Connect  338
download
  .NET Framework and SDK  62
  MQ Pub/Sub  61
  WebSphere Data Interchange import files  444
  WebSphere MQ CSD  59
  DTD
    import in WebSphere Data Interchange  440
    See document type definition
dual action PIP  697
DUNS (Data Universal Numbering System)  140, 714

E
e-business  4
eBusinessReady  695
ebXML  9
e-commerce  4
EDI  4
  and e-commerce  4
  ANSI X12  8
    sample message  372
  AS1  25
  AS2  25
  benefits  370
  business identifier in WebSphere BI Connect  141
  communication  373
  components  371
cXML  8
document exchange  15
document structure  371
download import files  444
ebXML  9
EDIFACT  8
  sample message  373
evolution  375
first era 7
fourth era 9
GENCOD 8
ISA segment in WebSphere BI Connect 141
message format standards 25, 371
message standards 371
messaging 15
network bandwidths 15
over the Internet 25, 375
over value-added network (EDI/VAN) 25
sample document 179
second era 8
solution components
batching 378
trading partner agreements 378
translator 377
solution elements 377
terms, concepts 368
third era 8
TRADACOMS 8
translation 53
UN/CEFACT 9
VAN 9, 374
VDA SEDAS 8
WebSphere Data Interchange for Multiplatforms 378
what is 25
xCBL 8
XML 8
XML based message formats 15
EDI (electronic data interchange) 367
EDI INT 375
EDI over the Internet
See EDI-INT
EDI over value-added network
See EDI/VAN
EDI solution message router 378
EDI standards 8
EDI/VAN (EDI over value-added network) 25
EDIFACT 8
what is 25
EDI-INT
AS1 25
AS2 25
what is 25
eif (export/import file) 441, 444
electronic business exchange 696
electronic data interchange
See EDI

electronic data interchange (EDI) 367
element 371
e-Marketplace 7
enable
B2B capabilities 155, 160, 270
digital signature 222
encryption 194
encryption 27, 190
certificate authority 194
enabling 194
export key 198
extract certificate 197
obtaining certificates 194
provide password for private key 199
Public Key Infrastructure 27
requirements for WebSphere BI Connect 193
RosettaNet 702
secret key cryptography 27
solving problems 356
update participant connections 216
upload
private key in WebSphere BI Connect 200
public certificate in WebSphere BI Connect 204
validating that it is enabled 218
versus digital signature 190
end-to-end validation 178
envelope 371
error event 514
event 514
creation standards 515
exception 514
classes 516
export/import file (eif) 441, 444
exporting a private key 198
external trading partner 411
extract certificate 197

F
failure notification 733
file system
gateway 149
target in WebSphere BI Connect 131, 172
fixed inbound workflow 631
fixed outbound workflow 631
Forward Translation table 480
framework 536
FTP 26
functionality in WebSphere BI Connect 282
security 319
what is 26
FTP Directory 284
FTP server
  access rights 310
  configuration 303
  configure 303
  create instance 305
  security 306
FTP target
  directory structure 286
  naming conventions 284

G
gateway 37, 123, 589
  assign to connection 166
  create
    file system 150
    FTPS 333
    HTTPS 157
    JMS 430
    creating 149
    label as default 152, 159
    match with partner’s target 158
    naming 251, 274
    parameters for JMS 430
    queuing 350
    See also sender
    what is 149
GENCOD 8
generating a self-signed certificate
  .NET 321
  ikeyman 194
getBusinessProcess() method 632, 637
  group
    define in AIX 102
    define in Windows 67

H
handler
  definition 491
  postprocess
    receiver 494
    sender 500
  preprocess
    receiver 493
    sender 500
  synch-check
    receiver 494
  handler chain 491
  hashing 27
    message digest algorithm (MD5) 28
    secure hash algorithm (SHA) 27
Healthcare Information Portability and Accountability Act
  See HIPAA
HIPAA 25
home directory 309
HTTP 26
  configuration in WebSphere BI Connect Express 253
  ports of WebSphere BI Connect 86, 119
  target in WebSphere BI Connect 129, 171
  view header 348
  what is 26
hubadmin 89

I
IBM WebSphere Data Interchange for Multiplatforms 378
ikeyman 194
import
  certificate in certificate store 325
  DTD in WebSphere Data Interchange 440
  private key in WebSphere BI Connect 199
  public certificate in WebSphere BI Connect 203
inbound fixed workflow 495
industry-oriented XML data structures 8
informational event 514
init method 639, 642, 645
init() method 544, 557, 630, 635
install
  database schema
    AIX 105
    Windows 73
  FTP server 302
  WebSphere BI Connect Express product 238
  WebSphere BI Connect product 81, 113
  WebSphere Data Interchange
    client 396
    server 389
integration broker solution 382
integration test 513
interaction 37, 123
create
binary 292
EDI 134
RosettaNet 725
XML 267
creating 132
validate action 727
what is 132
international XML data structures 9
interoperability testing 696
ItsoInboundFWProcessor 644

J
Java 29
JaxMe 598
JMS
change context 428
configure JNDI 427, 735
configure Pub/Sub
  AIX 103
  Windows 72
create
  context 428
  queue 429
  queue connection factory 429
create gateway 430
create target 432
structure of message 436
using JMSAdmin 429
validate configuration 435
verify installation
  AIX 98
  Windows 59
JMS gateway 430
JUnit 513

L
linear log 69, 102
log files
  FTP server 307
  WebSphere Application Server 339
  WebSphere BI Connect 88, 339
log on
  initial by community participant 157
  initial by participant 148
  WebSphere BI Connect 89, 140
  WebSphere BI Connect Express 242
logging information 73, 105

M
mailbox profile 400–401, 464
mapper 377
MD5 (message digest algorithm) 28
MDN
  error when MDN return address not specified 354
  not signed 358
  raw document 347
  request from partner 168
  request signed MDN from partner 226
  requested 168
  set attributes on connection 168, 226
menu options in WebSphere BI Connect 91
  hubadmin versus participant 151
message 24, 371
message digest algorithm (MD5) 28
message format standards 25
message integrity 220
message queuing (MQ) 23, 375
message router 378
message standard 371
messaging and queuing 23
metadata 29
MIME 28
MQ (message queuing) 23, 375
MQ Pub/Sub
  configure
    AIX 103
    Windows 72
  enable automatic start-up 71
  start broker 70
  verify
    AIX 99
    Windows 61
MQSeries queue profile 403, 465

N
network profile 401, 463, 478
NLS support 92
non-blocking mode 539
non-repudiation 220

O
OASIS 9
ODETTE 25
On Demand Business 4
Operator profile 38
Organization for Data Exchange through Teletransmission in Europe
See ODETTE
Organization for the Advancement of Structured Information Standards
See OASIS
outbound fixed workflow 498
overview
configuration process 123
digital signature 220
encryption and decryption 190, 192
management tools 338
WebSphere BI Connect - Express 236
WebSphere Data Interchange scenarios 420

P
package flow 132
package link to document flow definition 259
participant connection 156, 216, 589
activate 165
assign gateway 166
test 178
update for digital signature 226
update for encryption 216
update gateway 431, 434
Partner Interface Process (PIP) 696–697
support in WebSphere BI Connect 706
password
change 90, 149
in WebSphere BI Connect Express 243
for FTP gateway 294
for private key file 199
generated by WebSphere BI Connect 143
of key store 195
reset in WebSphere BI Connect 144
password of hubadmin 89
password policy 93
PIP (Partner Interface Process) 696
PKI (Public Key Infrastructure) 27
point-to-point application integration architecture 11
point-to-point solution 381
post-process handler 500
postprocess handler 494
preamble 702, 704
preprocess handler 493, 500
private key to own server 199, 211
problem determination delivery failure 341
incomplete AS2 setup 354
related to digital signature 358
related to encryption 356
view incomplete AS2 transactions 350
process flow 490, 510
process() method 630, 632, 635, 639, 643, 645
processResponse() method 546
profile for participant company 253
profile information 73, 105
profile management 38
properties
of a file system target 131
of a JMS gateway 430
of a JMS target 433
of an HTTPS gateway 158
of an HTTPS target 128
of community manager 143
of community participant 145
of self-signed certificate 196
protocol flow 132
public certificate to partner server 204
public key cryptography 27, 190
Public Key Infrastructure (PKI) 27
public/private key pair 194, 207

Q
queue 26
browse messages 319
define 712
queue connection factory 428
queue manager
add broker as service 71
create
AIX 102
Windows 68, 393
create listener 69
customize log 69
tune channel parameters
AIX 103
Windows 70
queuing 24

R
receiver 492–493, 537
collection point 493–494
configuration point 493–494
framework 540
processing flow 538
service level agreement 350
Service profile 406, 465
SHA (secure hash algorithm) 27
signature 220
signature verification 219
single action PIP 697
SMTP 26
  IMAP 26
  POP3 26
  what is 26
software prerequisites
  AIX 96, 98, 100
  Windows 58, 62
SSL (Secure Sockets Layer) 28
start
  MQ Pub/Sub broker 71
  WebSphere BI Connect components
    AIX 120
    Windows 88
  WebSphere BI Connect Express 241
startReceiving() method 545
stop
  document exchange process 353
  MQ Pub/Sub broker 71
  WebSphere BI Connect Express 241, 246
stopReceiving() method 545
stored procedures
  configure DB2
    AIX 101
    Windows 64
requirements
  AIX 100
  Windows 62
validate 80, 112
support
  download InfoCenter 338
  download SupportPacs 435
  download WebSphere MQ CSD 59
symmetric key algorithm 190
symmetric key encryption 27
synch-check handler 494
synchronous request 539

T
  target 36, 125
    create
      file system 172
      FTP receiver 289
    HTTPS 127
    JMS 432
    match with partner’s gateway 158
    naming 251
    properties for JMS 432
    See also receiver
    what is 127
technical directory 701
test connection 178
TPA (Trading Partner Agreement) 378
TRADACOMS 8
Trading Partner Agreement (TPA) 378
transaction 371
TransformationFactory 636
translator 377
transport development 536
transport header 549
transport protocol 26
  FTP 26
  HTTP 26
  SMTP 26
triggered queue 319
tune channel parameters
  AIX 103
  Windows 70

U
  UCC Solution Partner Program 695
  UCS 25
UDDI (Universal Description, Discovery, and Integration) 29
UN/CEFACT 9
unit test 513
United Nations Trade Data Interchange (UNTDI) 25
United Nations Trade Data Interchange Standards
  See UNTDI
Universal Description, Discovery, and Integration (UDDI) 29
Universal Multi-Octet Coded Character Set
  See UCS
UNTDI (United Nations Trade Data Interchange) 25
update usage of a private key 222
upload file to be sent in Express 278
upload private key 200
upload public certificate
  for certificate authority 223
  in WebSphere BI Connect 204
usage patterns for WebSphere Data Interchange
MQ Pub/Sub
   AIX 99
   Windows 61
software prerequisites
   AIX 96
   Windows 58
WebSphere MQ
   AIX 98
   Windows 59
VICS (Voluntary Inter-industry Communications Standards) 25
Voluntary Inter-industry Communications Standards (VICS) 25

W
warning event 514
WDIAdapter program 380, 387
Web services 29
   UDDI 29
   WSDL 29
Web Services Description Language (WSDL) 29
WebSphere BI Connect
   action 496
   activate participant connections 165
   architecture 32
   change password 90
   configuration point 491
   configure MQ resources
      AIX 103
      Windows 72
create
   community manager 139
   community participant 145, 257
   gateway 150
   file system 150
   for Express partner 271
   FTP 294
   FTPS 333
   HTTPS 157
   JMS 430
interaction
   binary 292
   EDI 134
   XML 267
target
   file system 130, 172
   FTP receiver 289
   HTTPS 127
JMS 432
XML format 264
database owner 73
default password 89
define group
  AIX 102
  Windows 67
define user IDs
  AIX 102
  Windows 66
development tool 511
document and event analysis 343
document flow definition 132
documentation 338
DUNS number 714
enable digital signature 222
enable encryption 194
event analysis 342
extending 55
features 32
FTP functionality 282
FTP target naming conventions 284
handler 491
handler chain 492
hubadmin user 89
implementing Advanced for AIX 47
inbound fixed workflow 495
inbound flow 125
install database schema
  AIX 105
  Windows 73
install product
  AIX 113
  Windows 81
key store password 195
label gateway as default 152, 159
log files, Windows 88
log on 89, 148, 157
outbound flow 123
overview of menu options 90
password policy 93
process flow 490
provided action 496
provided PIPs 706
receiver 492–493
receiver framework 494
requirements for digital signature 221
requirements for encryption 193
reset password 144
role-based configuration 122, 184
RosettaNet support 706
run-time parameters of components 86, 119
sender 499
sender framework 500
service level agreement 350
start components
  AIX 120
  Windows 88
stopping document exchange process 353
structure of JMS message 436
supported database products 74, 106
target 493
test connection 178
updating 536
upload
  private key 199
  public certificate 204
using FTP 51
validate 87, 120
  communication 178, 276
database schema
  AIX 111
  Windows 79
digital signature 231
  encryption 218
  JMS configuration 435
variable workflow 496
WebSphere BI Connect Enterprise
  implementing for Windows 46
WebSphere BI Connect Express
  automatic sending 279
  change password 243
  configure My Profile 256
  create participant 244
default password 242
  implementing for Windows 50
log on 242
  management tools 276
manual sending 277
  overview 236
start 241
stop 246
upload file for transmission 277
validate communication 276
WebSphere BI Connect header 549
WebSphere Data Interchange 367
  component view 399
  components
Mailbox profile 400
MQSeries queue profile 405
Network profile 401
partner profile 410
service profile 406
configure MQ resources 393
create
data transformation map 444
variable 459
XML dictionary 440
create database 391
development environment 386
download CSD 390
download import files 444
import DTD 440
installation and setup 389
installation server 389
installing the client 396
integration scenarios 420
operating modes 380
rules 413
run-time environment 387
target versus source mapping 446
usage patterns 381
XML support 380
WebSphere Data Interchange Client, creating a new system 398
WebSphere Data Interchange for Multiplatforms 378–379
WebSphere MQ
browse queue 319
client channel 751
configure trigger monitor 394
customize logging 69
download CSD 59
JMS, configuring JNDI 427, 735
setmqcap 103
structure of JMS message 436
tune channel parameters
AIX 103
Windows 70
using JMSAdmin 429
using runmqsc 712
verify level
AIX 98
Windows 59
WebSphere MQ-related artifacts 404
white box testing 513
WSDL (Web Services Description Language) 29

X
xCBL 8
XML
business identifiers in custom XML 266
cross-industry standards 9
cxXML 8
document definition 29
ebXML 9
growth of 8
import DTD in WebSphere Data Interchange 441
OASIS 9
RosettaNet 8
support in WebSphere Data Interchange 380
tag languages 29
UN/CEFACT 9
what is 29
xCBL 8
Version 4.2.2
WebSphere BI Connect
B2B Solutions using
B2B Solutions using WebSphere BI Connect Version 4.2.2

Implement EDI solutions using WebSphere BI Connect, WebSphere Data Interchange

This IBM Redbook introduces you to business-to-business (B2B) solutions based on IBM WebSphere BI Connect for B2B. In Part 1, you learn about B2B technologies and features, architecture and integration options of WebSphere BI Connect.

Part 2 describes the implementation of three editions of WebSphere BI Connect on Microsoft Windows and AIX. Within an environment of four trading partners, you learn step-by-step how to implement various B2B scenarios. You learn how to integrate an FTP server with WebSphere BI Connect so that you still have the visibility and manageability of WebSphere BI Connect. This part also demonstrates how to implement an AS2 exchange of electronic data interchange (EDI) documents and custom XML documents, as well as how to secure such exchanges via digital signatures and encryption.

Part 3 discusses the integration between WebSphere Data Interchange and WebSphere BI Connect. It shows you how the two products can work together in routing and transforming documents to and from trading partners.

Part 4 discusses how to implement various user exits that allow you to use a custom transport, custom packaging, and custom formats.

Part 5 explains how you can enable RosettaNet support in WebSphere BI Connect. It shows a way to test this support using standard utilities and viewers in WebSphere BI Connect.